AUTOCODE

un sistema simplificado

de codificación para la computadora

MERCURY

Instituto de Cálculo
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Noviembre 1961

NOTA PRELIMINAR

Hemos creído importante reunir en unas páginas las reglas esenciales para programar en el sistema de codificación automática AUTOCODE, atendiendo a la necesidad de los centros científicos e industriales de programar ellos mismos sus propios programas, facilitando así la utilización de la computadora Ferranti—Mercury instalada en el Instituto de Cálculo de la Universidad de Buenos Aires.

Incluimos dos apéndices que nos parecen de mucha utilidad al permitir mayor flexibilidad incluso a los no conocedores del sistema convencional.

Hemos de agradecer la paciente y minuciosa lectura de nuestra primera redacción, así como sus múltiples puntualizaciones, a la Dra. Cicely M. Popplewell. También agradecemos la valiosa colaboración de la Dra. R.Ch. de Guber.

Buenos Aires, octubre de 1961

E. García Camarero

del Instituto de Cálculo
de la Facultad de Ciencias
Exactas y Naturales

I. INTRODUCCION

1. Generalidades. El sistema autocode, como todos los sistemas de codificación automática, consiste en un programa traductor que vierte un lenguaje simbólico, (el lenguaje autocode), muy próximo al lenguaje matemático, en el lenguaje absoluto de la máquina. Una vez hecha la traducción, comienza la ejecución del programa absoluto, uno de cuyos pasos será la lectura de los datos numéricos del problema y la impresión de los resultados.

El programador en autocode no precisa necesariamente, conocer el lenguaje absoluto, ni los detalles funcionales de la Mercury, sino solamente el lenguaje autocode y la disposición funcional de la máquina ideal sobre la que trabaja, lo que le permite atender sus necesidades de cálculo numérico sin estar próximo a la computadora que se encargará de resolverlo.

2. La computadora. Nuestra máquina ideal como todas las máquinas calculadoras digitales automáticas, consta de las siguientes partes (fig.l):

a) una memoria
b) una unidad aritmética
o) una unidad de control
d) dispositivos de entrada y salida

La unidad aritmética de nuestra "máquina ideal" es muy potente pues además de las cuatro operaciones elementales, es capaz de efectuar una serie de funciones de bastante complejidad.

La unidad de control es el centro encargado de interpretar todas las sentencias y ejecutar aquéllas que son netamente de control o lógicas.

Los dispositivos de entrada y salida nos permiten introducir dentro de la máquina el programa y los datos y obtener los resultados perforados en una cinta de papel o impresos en la forma que nos convenga.

La memoria es el lugar en donde se almacenan el programa, los datos y los resultados durante el tiempo de ejecución.

Así como para el programador de autocode, no le es importante conocer la estructura interna de los dispositivos de entrada y salida ni de las unidades de control y aritmética, le interesa aclarar algunos puntos sobre la memoria.

La memoria es un depósito en donde pueden ser almacenados números e instrucciones. Está dividida en numerosos compartimentos, en cada uno de los cuales se puede depositar un número o una instrucción. Para individualizar estos

compartimentos están numerados; su número se llama dirección (señas o ubicación). Es decir que a cada compartimento o casilla se le asocian dos números, uno su dirección que sirve para individualizarlo y otro su contenido que puede ser una instrucción o un dato numérico. Como una misma dirección puede contener números distintos en instantes diferentes, las direcciones juegan un papel análogo al de las variables en álgebra; es decir que podemos operar con direcciones y las instrucciones tomarán los contenidos, igual que en álgebra operamos con letras que deben ser sustituidas por sus valores cuando buscamos resultados numéricos.

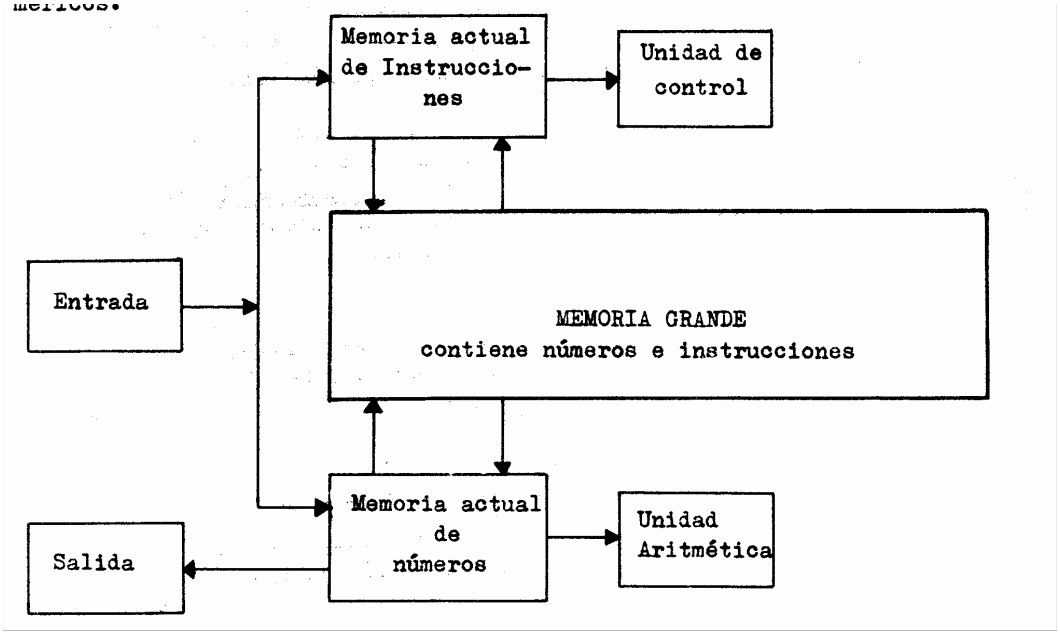


Fig. 1. — Diagrama funcional de la computadora MERCURY para los efectos del sistema de autoprogramación AUTOCODE.

Para programar en autocode, vamos a suponer la memoria dividida en tres partes o la existencia de tres memorias diferentes. La memoria actual dividida en dos partes una para instrucciones y otra para números, y la memoria grande

La memoria actual de instrucciones, está constituida por núcleos de ferrita de acceso muy rápido. Cuando un programa es demasiado largo necesitamos dividirlo en partes, llamadas capítulos, la longitud de un capitulo está limitada por la capacidad de la memoria actual de instrucciones. La memoria actual de números, también de núcleos de ferrita, tiene capacidad para 521 variables como indica la fig. 2 (véanse mas adelante las secciones dedicadas a variables e índices).

La memoria grande, constituida por un tambor magnético de acceso más

lento, tiene capacidad para contener varias veces la memoria actual.

Solamente son obedecidas las instrucciones almacenadas en la memoria actual, por eso para programas de más de un capitulo es preciso utilizar la técnica de trasladar de vez en vez de la memoria grande a la actual el capitulo de nuestro programa que debe ser obedecido en aquel momento, con las variables y constantes correspondientes. Esto justifica el nombre con que hemos designado la memoria actual.

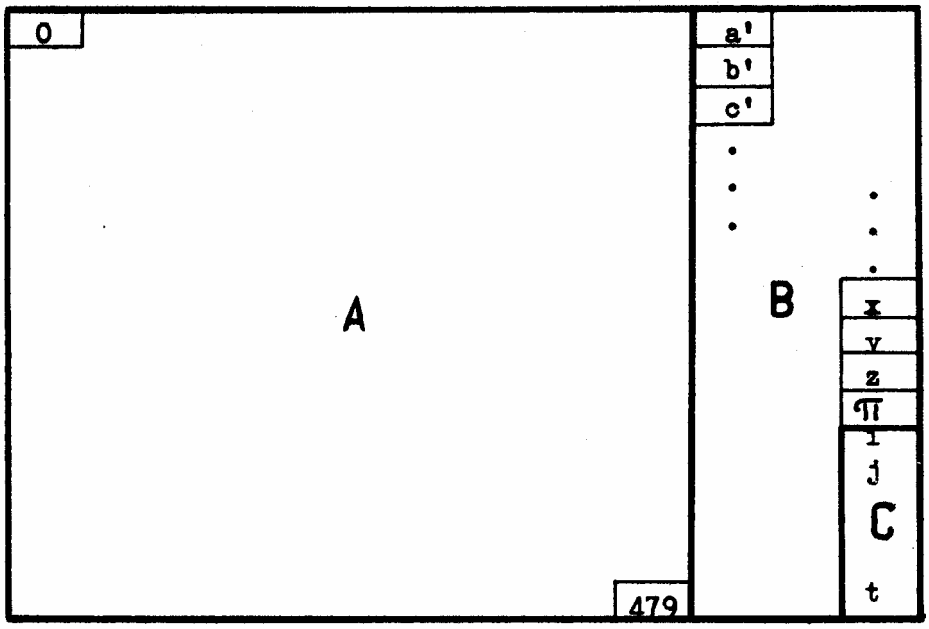


Fig.2. — Memoria actual de números. Están indicadas con A, B y C las zonas dedicadas a las variables especiales, e índices, respectivamente. (Ver con detalle en Apéndice I).

3. El lenguaje autocode. El lenguaje autocode está constituido por sentencias. La máquina traduce las órdenes contenidas en cada sentencia y las realiza una vez traducido todo el programa. Las sentencias constan fundamentalmente de una parte operativa y de unos operandos. Estas sentencias, verdaderas oraciones gramaticales del lenguaje en que estamos trabajando, son de diverso tipo y están constituidas por distintas partes.

Atendiendo a su función operativa las podemos clasificar en:

— Instrucciones aritméticas.
— Instrucciones de control.
— Instrucciones de entrada y salida.
— Directivas.

Las partes esenciales de que puede constar una sentencia son:

— variables
— índices
— números
— signos aritméticos
— signos especiales
— palabras.

En todos los casos los elementos primarios son letras, cifras y signos; el conjunto de todo esto forma nuestro alfabeto, que es el siguiente:

a b c d e f g h i j k l m n o p q r s t u y w x y z π
· 0 1 2 3 4 5 6 7 8 9 + – = ≠ > ≥ ≈ (,) → * / ψ ‘ ?

Queremos hacer notar que así como es importante la sintaxis de nuestro lenguaje, es decir el conjunto de reglas que debemos seguir al escribir nuestras sentencias, es también de capital importancia la ortografía, es decir escribir exactamente cada una de las partes de la sentencia en su forma correcta, ya que el cambio de una letra por otra o la simple omisión de uno de los signos de nuestro alfabeto, puede producir efectos distintos a los deseados, o la parada de la máquina por no saber interpretar nuestra sentencia.

NOTA: Los caracteres impresos por loe equipos actualmente instalados en el Instituto de Cálculo de la Universidad de Buenos Aires, son los siguientes:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z £
• 0 1 2 3 4 5 6 7 8 9 + – = ≠ > ≥ ∪ (,) → * / x π ?

debiendo observar las especiales representaciones de π ≈ ‘ ψ

II . ELEMENT0S DE LAS SENTENCIAS

1. Variables. Como ya hemos indicado, las casillas de la memoria pueden almacenar números distintos en instantes diferentes, y las direcciones de estas casillas juegan un papel análogo al de las variables en álgebra. Por eso en le lenguaje autocode llamamos variables a las direcciones de las casillas don donde se pueden almacenar los distintos números que utilizamos durante los cálculos. Tenemos dos tipos de variables: variables actuales y variables auxiliares. Las primeras ocupan lugares de la memoria actual de números y pueden ser hasta 509; las segundas están ubicadas en la memoria grande pudiendo alcanzar hasta 10.752 lugares .

1.1 Variables actuales. Las 509 variables actuales de que disponemos, se dividen en dos grupos: variables principales y variables especiales.

a) Variables principales.

Hay 480 variables principales que pueden ser designadas por una de las siguientes letras

a b c d e f g h u v w x y z π

afectadas de un subíndice

Ejemplos: a_0 v_j h_{32} z_r

→

Previamente a la utilización de las variables principales en el programa, hemos de reservar las casillas de memoria que representan; esto se hace utilizando el símbolo → de la siguiente forma:

$a \rightarrow \alpha$
$b \rightarrow \beta$
$c \rightarrow \gamma$
. . . . . .

lo que tiene por efecto reservar las posiciones 0 a α para las variables a_0 $a_1 \ldots a_\alpha$, las posiciones $\alpha + 1$ a $\alpha + \beta + 1$ para las variables b_0 $b_1 \ldots b_\beta$, las posiciones $\alpha + \beta + 2$ a $\alpha + \beta + \gamma + 2$ para las variables c_0 $c_1 \ldots c_\gamma$, y así sucesivamente.

Ejemplo: Las expresiones

$x \rightarrow 100$

$y \rightarrow 36$

$z \rightarrow 1$

reservan las posiciones 0 a 100 para las variables x_0 x_1 . . . x_{100}, las posiciones 101 a 137 para las variables y_1 y_2 . . . y_{36}, y las posiciones 138 y 139 para las variables z_0 y z_1 .Si necesitamos una sola variable utilizamos las variables especiales que tratamos en el apartado siguiente.

b) Variables especiales.

Estas variables son 29 y están representadas por las siguientes letras sin subíndice

a' b' c' d' e' f' g' h' u' v' w' x' y' z'

a b c d e f g h u v w x y z' π

a las que corresponden las posiciones fijas de la memoria actual comprendidas entro 480 y 508 respectivamente. En la posición $\pi = 508$, mientras no lo modifiquemos, está almacenado el numero 3.141 592...

Las variables con acento se reservan generalmente para casos especiales. No existe la variable π'

1.2 Variables auxiliares. Cuando no son suficientes las 509 variables de que disponemos en la memoria actual, es preciso utilizar las variables auxiliares, que son posiciones determinadas de la memoria grande. Estas variables deben ser trasladadas a la memoria actual para su uso. A cada una de estas posiciones se le asigna un número, su dirección, que varia de 0 a 10751. No siempre podemos utilizar las 10752 casillas o variables auxiliares disponibles ya que las últimas 512n casillas están ocupadas por los capítulos l, 2, 3...n de nuestro programa. (Véase IV.3 y Apéndice 1).

2. Índices. Son las direcciones de las 12 últimas posiciones de la memoria actual de números y las designamos por las letras

i j k l m n o p q r s t

Estas posiciones pueden sólo albergar números enteros comprendidos en el intervalo (−512 , 511). Se utilizan principalmente como subíndices.

Ejemplo: si tenemos la variable a_i y el contenido del registro i es 3, la anterior variable equivale a a_3

3. Números. Los números son la materia prima sobre la que actúan las instrucciones y son alojados en las casillas reservadas por las variables o expresados

en forma explicita en el programa. A los números incluidos en forma explicita en el programa los llamaremos constantes, y a los que deben ser leídos durante el proceso de cálculo, datos. Las constantes ocupan casillas de la memoria actual de instrucciones. La memoria actual de números está reservada para los datos, para los resultados parciales y para las soluciones antes de su impresión. Aunque la máquina trabaja siempre a punto flotante con números contenidos en el intervalo $10^{-70} < | X | < 10^{-70}$, los datos pueden ser leídos en punto decimal fijo, y casi siempre son impresos de esta manera. Las constantes deben ser escritas siempre en la forma de punto decimal fijo.

3.1 Punto decimal fijo. <SIGNO> PARTE ENTERA) <PUNTO> <PARTE FRACCIONARIA>
La forma de punto decimal fijo es la habitual de los números decimales, es decir están formados por el signo, las cifras de la parte entera, el punto y las cifras de la parte decimal. En autocode podemos omitir el signo si es positivo, los ceros no significativos y el punto el fuese un número entero. Delante del punto puede haber un número cualesquiera de cifras, pero la parte fraccionaria puede contener 24 cifras como máximo. En todos los casos la máquina opera solo con 108 diez dígitos más significativos.

Ejemplos: +3.570 o 3.57
– 0.327 o –.327
+321.00 o 321

3.2 Punto decimal flotante. <MANTISA> <COMA> <EXP0NENTE>
La forma decimal flotante, es la escritura abreviada de expresiones del tipo

$$\pm \alpha \times 10 \pm \beta$$

(a α lo llamamos mantisa y a β exponente) que expresamos simplemente

$$\pm \alpha \ , \ \pm \beta$$

El exponente debe ser entero y estar comprendido en el rango (–128,127). En todos los casos deben estar comprendidos en el rango:
2-256 <X<.. 2256

Ejemplos: –27,3 expresa -27×10^{3} o –27000
+0.5, –2 expresa 5×10^{-3} o 0.005

4. Signos aritméticos. Para expresar las cuatro operaciones aritméticas fundamentales empleamos los signos o los convenios siguientes:

a) Producto . <YUXTAPOSICION DE LOS FACTORES>
Para expresar el producto de varios factores yuxtaponemos uno detrás de otro los distintos factores que intervienen en el producto sin que los separe ningún signo. Los factores pueden ser variables, índices o constantes, pero no expresiones algebraicas

entre ellos.

Ejemplos: $3abb$ $4.2xy$ x_ix_i u_iv_jwk
$5ay$ $2iz_j$ $6jz$ $6z_j$
ijk $mna_m\ b_n$

No son licitas expresiones como las siguientes:

$a\ (b + c)$

Como veremos más adelante expresiones del tipo

$6zj$ serán interpretadas como $6z_j$.

b) Suma

–	+

Los signos + ó – entre dos términos tienen por efecto producir la suma o diferencia de dichos términos. Los términos pueden ser variables, índices y constantes, o el producto de varios de ellos.

Ejemplos: $3+v_i$ $x+j+k-4.5$ $jk + pq$
$ab+2$ $a'-b'b'$ $2x_i+.25y_j-327.6$

c) Cociente /

/

El signo / colocado entre el dividendo y el divisor tiene por efecto realizar el cociente del primero por el segundo. El dividendo debe ser una variable, un índice, o una constante, o el producto de algunos de ellos, pero no una expresión en la que intervengan los signos + 6 −. El divisor ha de ser una variable, un índice, o una constante, pero no el producto o suma de alguno de ellos.

Ejemplos: a/b $3xy/z$ $3if_k/f_j$
$5.21/m$ v/q $a'b'/c'$

Expresiones del tipo

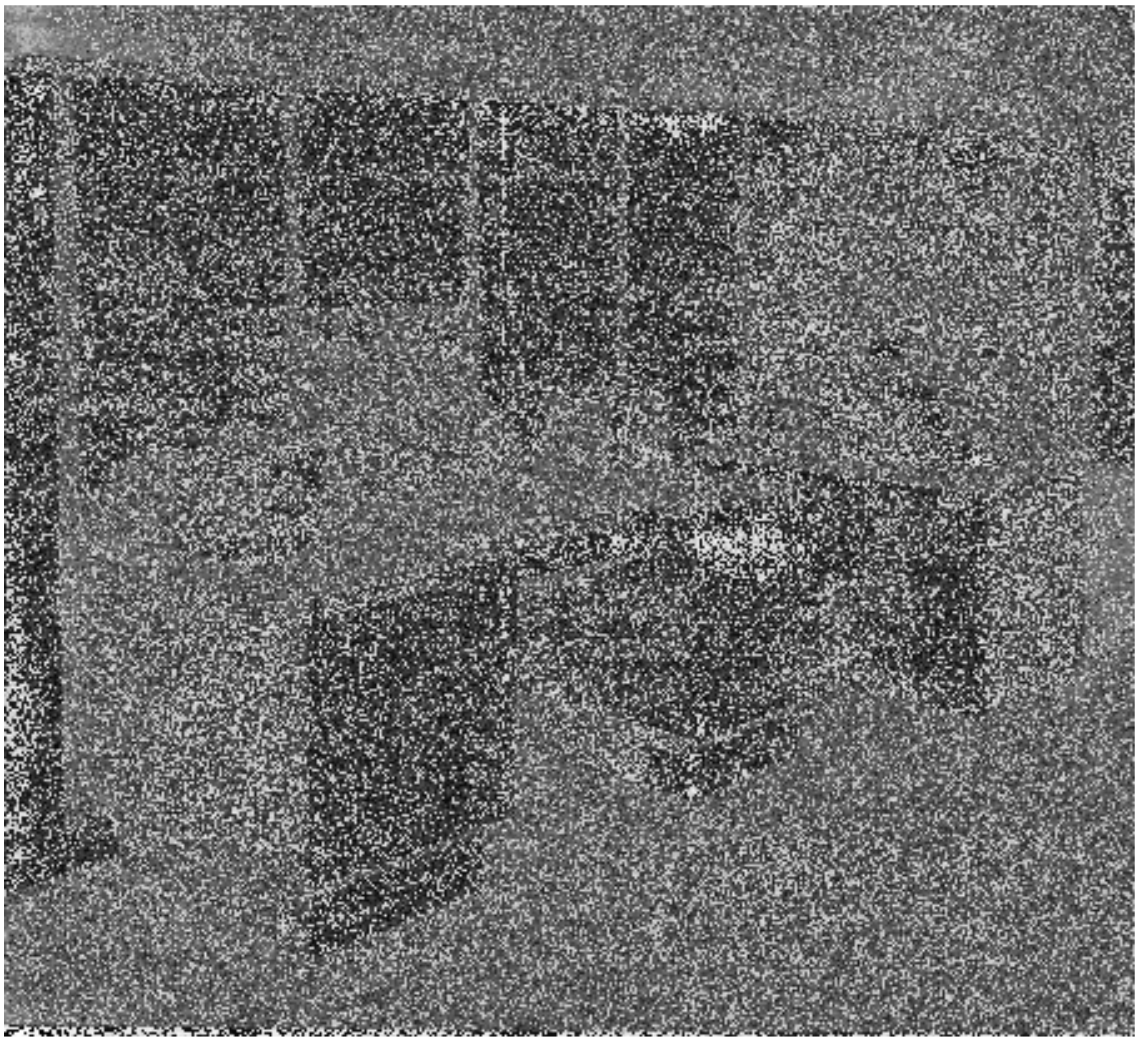
$a+b/c+d$

el sistema autocode lo interpreta como

$a+(b/c)+d$

5. Signos especiales y palabras. Además de las partes de las sentencias vistas hasta aquí, existen signos especiales y palabras cuya función es facilitar la expresión de las sentencias. Aunque su empleo es casi siempre análogo al habitual en el 1enguaje matemático, o tiene mucha analogía con él, no siempre es así, como ocurre con el uso de los paréntesis que nunca se pueden emplear para asociar varios sumandos en un factor, o con la coma que nunca expresa la separación de la parte entera y fraccionaria de un número, o con el punto que nunca expresa el producto. Una palabra puede expresar toda una sentencia, como ocurre

con algunas instrucciones de control o con las directivas, o con el nombre de una función.



Consola de la Mercury, donde se ve el lector fotoeléctrico y el perforador de salida. Los resultados pueden ser impresos en una teleimpresora ubicada en un escritorio lateral, a la derecha.

III. LAS SENTENCIAS

Las sentencias del lenguaje autocode son las que dan sentido a los símbolos estudiados en el apartado anterior que aislados (salvo en el caso de palabras sentencias) carecen de toda significación para la máquina. Las sentencias según su función en el programa, se pueden clasificar, como ya hemos indicado, en:
Instrucciones aritméticas, instrucciones de control, instrucciones de entrada-salida y directivas.

1. Instrucciones aritméticas. Las sentencias aritméticas se caracterizan por la la presencia de un signo = (≈), que produce el almacenamiento del número obtenido o contenido en el segundo miembro, en la dirección indicada primero. El primer miembro puede ser únicamente una variable, general o especial, o un índice, cada uno de estos dos casos determina la forma del segundo miembro. La longitud de una sentencia puede ser de hasta 68 caracteres (inc1uyendo espacios).

1.1. El primer miembro es una variable. Cuando el primer miembro es una variable, el segundo miembro puede tomar una de las siguientes formas:

a) el segundo miembro es una expresión algebraica en donde pueden intervenir las cuatro operaciones aritméticas elementales entre variables, índices, y constantes.

Ejemplos: y =2x+3.5z/2–i

w_s =3.5jx–4.6v_0 w_i/s –.03272+$z_{(s-2)}$ + i

Hemos de notar que un índice o un entero escrito después de una variable será considerado como un subíndice, ya que nuestra máquina lee todos los caracteres a un mismo nivel, así el segundo ejemplo anterior será escrito para la máquina en la forma:

ws = 3.5jx−.6v0wi/s –.03272 + z (s–2) + i

Este ejemplo pone también de manifiesto la necesidad del paréntesis cuando aparecen operaciones aritméticas con los subíndices Se aconseja escribir los factores de cada término de una expresión algebraica en el siguiente orden: constante, índices, variables.

b) el segundo miembro es una sola función de una variable de las que siguen a continuación:

y = ψ sq rt (x)	x >0
y = ψ sin (x)	x en radianes
y = ψ cos (x)	x en radianes
y = ψ tan (x)	x en radianes
y = ψ exp (x)	e^x x<177
y = ψ log (x)	$\log_e x$ x>0
y = ψ mod (x)	/x/
y = ψ int pt (x)	parte entera de x
y = ψ fr pt (x)	parte fraccionaria de x
y = ψ sign (x)	y = ± 1
y = ψ poly (x) a_0, n	a_0 es la dirección del primer coeficiente y n el grado del polinomio.
Y = ψ parity (n)	$y = (-1)^n$

El argumento de las funciones anteriores puede ser una variable, un índice o una constante o una expresión algebraica entre ellos, excepto en la función parity que debe ser un índice o una expresión algebraica entera entre índices y enteros.

Ejemplos: y = ψ sq rt (xi + 2yj)

Y = ψ sin (π z/180)

c) el segundo miembro es una de las funciones de dos variables que siguen a continuación:

z = ψ divide (x, y)	x/y
z = ψ arc tan (x, y)	arctg (y/x)
z = ψ radius (x, y)	$\sqrt{x^2 + y^2}$

los dos argumentos pueden ser variables, índices o constantes, o expresiones algebraicas entre ellos.

Ejemplos: z = ψdivide (3ixj + 2iyj , 5i/j)

z = ψ radius (a + b , 6.28)

Queremos insistir aquí nuevamente sobre lo importante de la ortografía en autocode Particularmente para las funciones advertimos la necesidad de poner la letra griega ψ inmediatamente detrás del signo **=,** nombrar las funciones con las mismas letras dadas en la lista, y encerrar el argumento entre paréntesis.

Hemos de notar que nuestra máquina en todas las instrucciones aritméticas caracterizadas por el signo **=,** de primer miembro variable redondea los resultados de las operaciones suma, resta, multiplicación y división, así como la lectura de números no enteros.

Si en una instrucción aritmética de primer miembro variable utilizamos el signo ≈ en lugar de utilizar **=,** las operaciones suma, resta y multiplica

ción se efectúan sin redondeo.
Hemos de notar que las funciones $y = \psi$ int pt (x), $y = \psi$ fr pt (x), $y = \psi$ mod (x), $y = \psi$ parity (n), $y = \psi$ sign (x), no son nunca redondeadas, ni tampoco expresiones de la forma y = a. Todas las demás funciones son siempre redondeadas. Si el argumento de las funciones es una expresión algebraica, será calculado con o sin redondeo según que el primero y segundo miembros estén separados por los signos **=, ≈.**

1.2 El primer miembro es un índice. En el caso en que el primer miembro es un índice, el segundo puede tomar una de las siguientes formas:

a) un polinomio entero entre índices y números enteros comprendidos en el intervalo (−512 , 511)

Ejemplos: $i = 2j + j$
$m = rs - st$
$j = i + 2$

No son válidas las expresiones como:

$i = v + w$
$j = i + 3.121$
$k = m/n$

b) una de las funciones indicadas a continuación

$i = \psi$ int pt (x) i–parte entera de x
$i = \psi$ max (x_0, m, n) $m < n$
$i = \psi$ min (x_0, m , n) $m < n$

La x en el primer caso puede ser cualquier variable, o alguna expresión algebraica entre variables, índices y constantes. En los casos segundo y tercero i expresa el subíndice del mayor o menor número de la sucesión $x_m x_{m+1} ... x ... x_n$, extraída de la

$x_0 x_1 ... x_p$, $p \geq n$. En el caso de haber varios máximos o mínimos iguales elige el de índice menor.

Ejemplos: $i = \psi$ int pt (3x−y)
$I = \psi$ min (v_0, 7, 21)

Nunca hay redondeo en las instrucciones aritméticas con primer miembro índice.

2. Instrucciones de control. Son aquellas instrucciones que nos permiten llevar el control de nuestro proceso de cálculo, es decir mediante ellas organizamos la secuencia lógica de nuestros programas, ordenamos la entrada y salida de datos,

o nos permiten llevar un control externo de la marcha de nuestro proceso. Esta sección la dividiremos en cinco partes: Ciclos, Saltos, Salto entre capítulos, Entrada-Salida datos, e Instrucciones de control externo.

2.1 Ciclos. Generalmente en los procesos de cálculo se presentan grupos de operaciones que hemos de repetir varias veces con datos distintos, a lo que llamamos ciclos. Para efectuar un ciclo el sistema autocode posee el siguiente par inseparable de instrucciones:

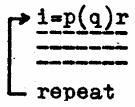
```
i = p (q) r
repeat
```

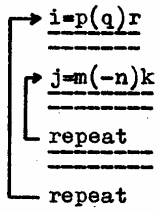
La primera instrucción, que podemos llamar cabeza del ciclo, puede tomar una de las siguientes formas:

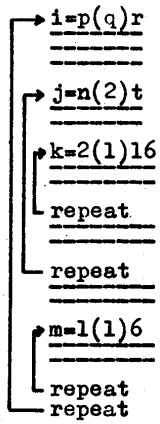
$$i = p\ (q)\ r \quad o \quad i = p\ (-q)\ r$$

e indica que el ciclo que comienza en la siguiente instrucción y termina con la instrucción repeat, se efectuará dando al índice i el valor inicial p, y el valor final r, pasando del uno al otro mediante pasos de valor q o −q, según los casos; alcanzado el valor r, no considerará la instrucción repeat y continuará secuencialmente. Las letras p, q, r pueden ser índices o enteros positivos comprendidos en el rango (0 , 511). Debe verificarse que r − p sea un múltiplo de q. Los números p, q, r han de ser positivos.

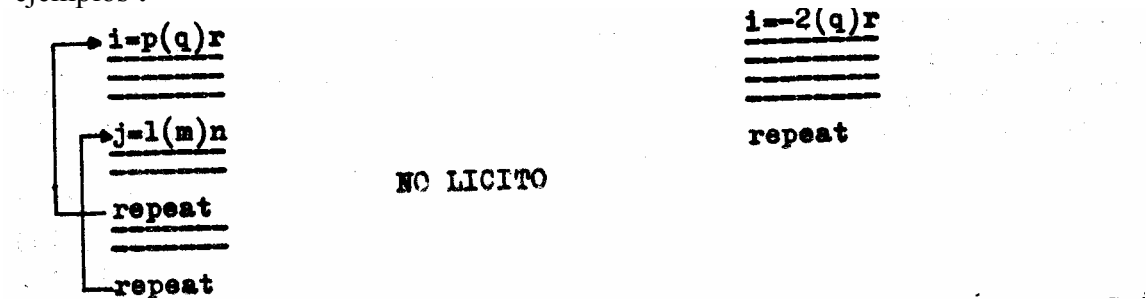
Dentro de un ciclo pueden existir otros ciclos, con la limitación que una instrucción determinada puede estar contenida en ocho ciclos, como máximo, es decir: no se pueden superponer más de ocho ciclos. También hemos de notar que a cada cabeza de ciclo, le corresponde un único repeat. Los siguientes esquemas exponen los distintos casos:







Hemos de notar que en el último ejemplo, aunque dos ciclos terminan con la misma instrucción, necesitan dos veces la palabra repeat, ya que cada cabeza de ciclo necesita su correspondiente repeat y viceversa. No son válidos los siguientes ejemplos :



2.2 Saltos. Cuando rompemos el orden secuencial de nuestro programa, es decir, en un determinado punto continuamos con una instrucción distinta de la siguiente, decimos que hemos efectuado un salto. Para efectuar un salto precisamos de una instrucción que lo ordene, y de una marca que nos individualice la sentencia a la que queremos llegar. En la presente sección definiremos las marcas, y los tres tipos de instrucciones de salto: salto incondicional, salto condicional, camino múltiple.

n)

a) Marcas

Cuando a una sentencia se puede llegar por un camino distinto del secuencial es preciso identificarla, para lo cual se usan las marcas. Estas consisten en un entero positivo del rango (1,127) seguida de un paréntesis cerrado y colocados (número y paréntesis) delante de la sentencia.

Ejemplos: 3) y = x + 5b

32) i = 6

14) z – radius (x , y)

jump n

b) Salto incondicional

La instrucción jump n, donde n es un entero positivo menor que 128, produce el salto a la sentencia numerada con la marca n). Este salto se puede verificar hacia adelante o hacia atrás indistintamente.

Ejemplos: jump 3 4) ——

——— ———

——— ———

3) ——— jump 4

jump n, α, σ, β

c) Salto condicional

Esta instrucción produce el salto a la instrucción marcada con la marca n), siempre que se verifique la relación σ entre α y β.

La letra σ expresa cualquiera de los símbolos = ≠ > ≥ , y las letras α y β pueden ser ambas variables, o ambos índices, pueden ser también variable y constante, o índice y entero, pero no es posible comparar una variable con un índice directamente,

Ejemplos: jump 3, $a' > a_i$
jump 127, $b \neq i$
jump 42, $a_i > 0.01$
jump 71, $0.001 = a_i$
jump 26, $n > k$
jump 5, $n \neq 1$
jump 1, $3 \geq k$
jump 102,25=n

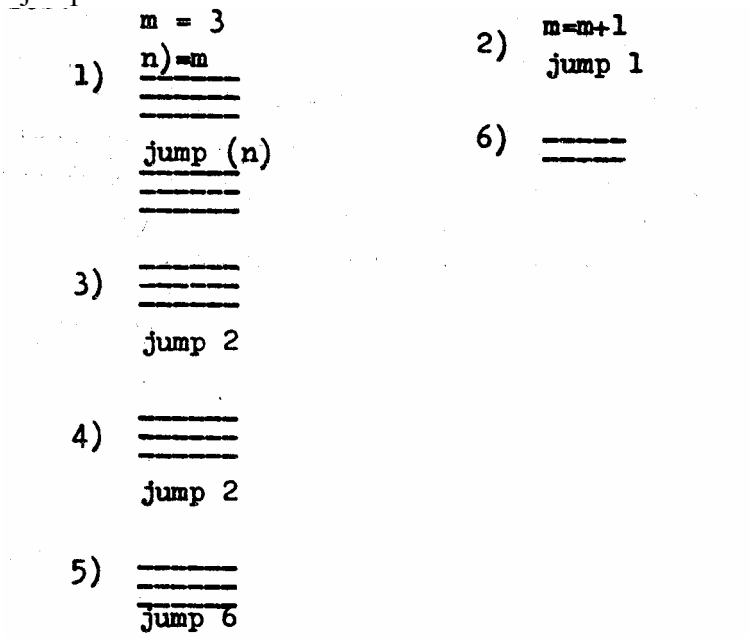
Hacemos notar que no es conveniente condicionar un salto por la expresión = ó ≠ entre dos variables, dada la improbabilidad de que esa igualdad ocurra exactamente a lo largo de un proceso de cálculo.

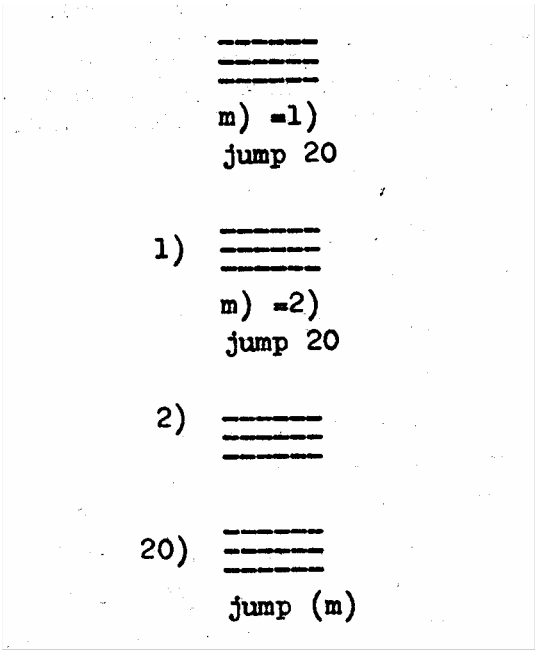
```
n) = m)
jump (n)
```

d) Camino múltiple

La instrucción jump (n) produce un salto condicionado al valor de la marca n) (n siempre un índice) que ha tenido que ser definido con anterioridad mediante la instrucción n) = m) (m es un entero).
Dado que la instrucción n) = m) es muy lenta (17 ms) convendrá emplearla sólo en casos imprescindibles. Mucho más rápida es la instrucción n) = entero) (120 μs).

Ejemplos:





2.3 Salto entre capítulos. Si antes de concluir un capitulo necesitamos saltar a una instrucción contenida en otro capitulo debemos utilizar las siguientes instrucciones especiales de control: across m/c, o el par inseparable down m/c, up.

across m/c

a) Ida (cambio simple)

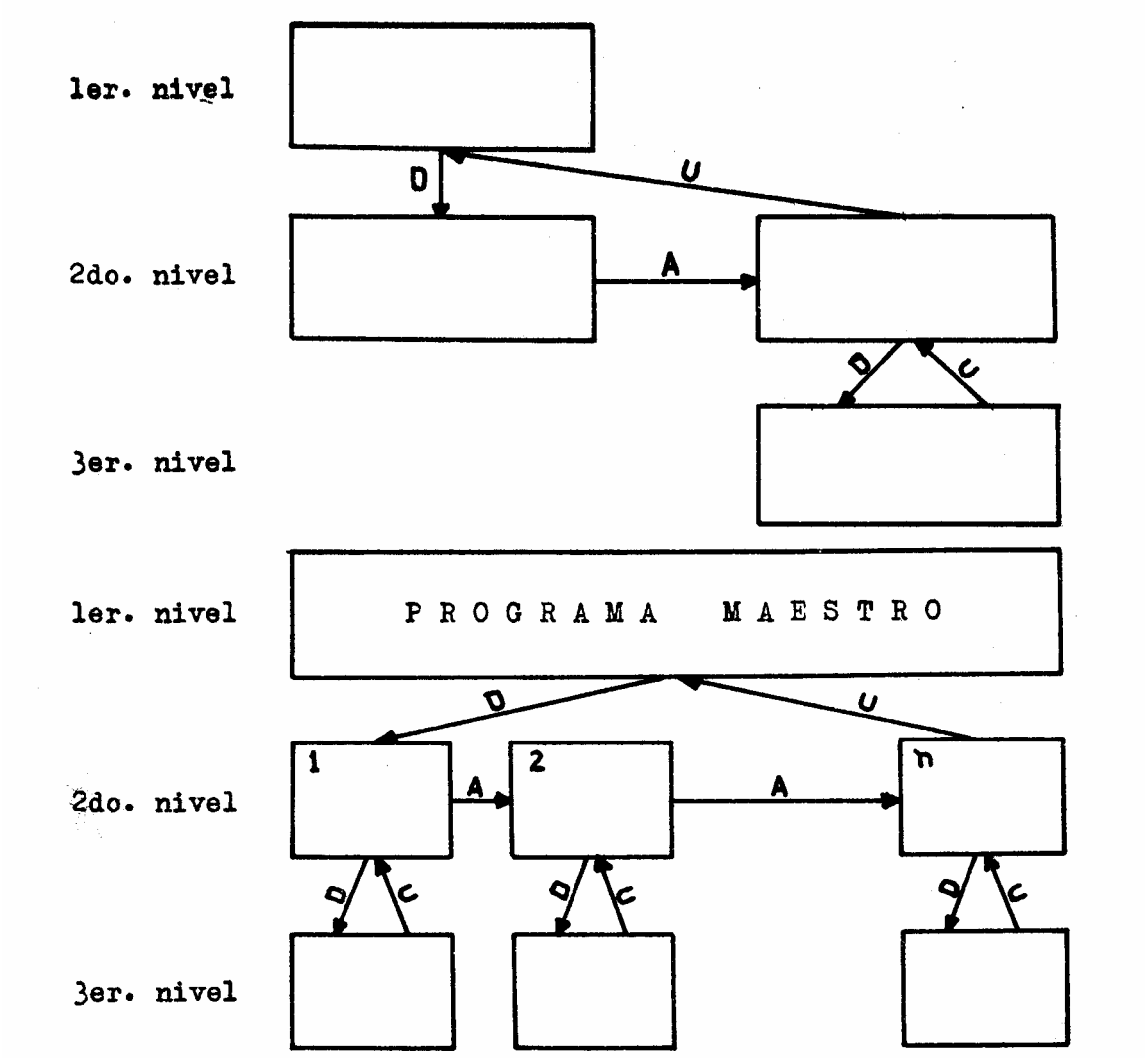
Esta instrucción tiene por efecto continuar los cálculos en la instrucción de marca m) contenida en el capitulo número c (véase la sección directivas). Para lo cual deposita el capítulo c en la memoria actual y comienza a ejecutar el programa en la marca m).

down m/c up

b) Ida y vuelta (cambio con regreso)

Son dos instrucciones inseparables, la primera tiene el mismo efecto que across m/c, con la particularidad de que cuando en la ejecución del programa encuentra la instrucción up, continúa en la instrucción siguiente al down m/c de donde partió.

Los siguientes diagramas ilustrarán sobre algunas posibilidades:



3. Instrucciones de control externo. Así llamamos a una serie de instrucciones destinadas a ayudar a controlar nuestro proceso de cálculo mediante señales sonoras. Unas nos indican aproximadamente en qué parte del programa nos hallamos y otras nos avisan cuándo hemos de hacer alguna instrucción manual o cuando hemos terminado los cálculos. Las que vamos a describir son: Hoot, Halt, End.

Hoot

a) Señal

Esta instrucción tiene por efecto producir una señal sonora en el altavoz de un segundo de duración. De esta manera podemos jalonar nuestro proceso de cálculo y saber en cada momento si el programa opera correctamente.

Halt

b) Alto

Mediante esta instrucción entramos en un ciclo que consta de las siguientes partes:

– producción de la nota sonora a
– producción de la nota sonora b
– leer en la consola
– si en la consola no hay determinada información continuar el presente ciclo.
– el en la consola hay determinada información continuar el proceso de cálculo.

Esta instrucción nos avisa que hemos llegado a un punto en que hemos de operar manualmente. (Véase Apéndice II).

end

c) Fin
Esta instrucción situada al final de nuestro programa nos indica de forma sonora que hemos terminado los cálculos y no es posible continuar.

4. Instrucciones de entrada y salida. Necesitamos unas instrucciones para que nos ordenen la entrada y salida de datos, es decir para almacenar en la memoria los datos que vamos a utilizar, o para imprimir sobre un papel los resultados. Las instrucciones principa1es son read y print. Además existe otra instrucción de salida condicional expresada por **?** y otras instrucciones auxiliares para permitir ordenar en forma de tabla los números impresos, éstas son newline y space ; así como otra rmp que nos permite leer un programa durante la ejecución de los cálculos.

4.1 Entrada- Tenemos dos instrucciones, read (α) y rmp que nos permiten leer una cinta colocada en el órgano de entrada, según se trate de cinta de datos o cinta de programa, respectivamente.

Read (α)

a) Leer datos
Mediante esta instrucción almacenamos en la dirección α, que puede ser una variable o un índice, el próximo número perforado sobre la cinta que está en el órgano lector. Estos números pueden estar perforados en la forma de punto fijo o de punto flotante si α es una variable, pero si es un índice debe ser perforado corno un entero comprendido en el intervalo (–512 , 511). Es importante que los números estén perforados en el orden requerido.
Ejemplos: Si en el órgano lector tenemos el número 2.2575, la instrucción

read (a_0)

tiene por efecto almacenar en la casilla de dirección a_0 el número 2.2575, después de transformarlo en punto flotante.

r p m

b) Leer más programa

La instrucción rmp (iniciales de read more programme) tienen por efecto leer y traducir la cinta de programa situada en el órgano de lectura. Para que continúe automáticamente el proceso de cálculo, el programa recién leído debe incluir un capitulo 0, que destruirá el capitulo 0 existente.

4.2 Salida. La salida de información está regulada por las siguiente instrucciones :

print (α) m, n

a) Impresión de resultados

En la instrucción print (α) m, n, indicamos por m y n índices o enteros, y por α a una variable, un índice, una constante, o una expresión algebraica entre ellos. Si $m \neq 0$, esta instrucción produce la impresión de α en punto decimal fijo, con tantos lugares enteros como indica m y decimales como n; si n = 0 no imprime el punto decimal. Si m = 0, $\alpha \geq 10^{14}$, imprime α en forma de punto decimal flotante, utilizando tres lugares para el exponente. Si α es un índice su impresión contendrá como parte fraccionaria tantos ceros como indica n. Si el número α tuviese parte fraccionaria imprimirá su valor redondeado en la última cifra .

Ejemplos: Si en a_0 hay el valor –3.27721675 la instrucción

print (a_0) 2,5

imprimirá

−3.27722

newline
space

b) Tabuladores

Mediante las palabras space y newline efectuamos las operaciones de dejar un lugar en blanco y de cambiar de línea en nuestro escrito, sobre el teletipo. Hemos de tener en cuenta cuando queramos imprimir nuestros resultados en forma de tabla, que la línea del teletipo tiene solamente 68 espacios, y que detrás de cada numero la máquina deja automáticamente dos espacios, así que en el caso general cada número ocupa m + n + 4, ó m +3 si n = 0. Si escribiera en la forma de punto decimal flotante ocuparía n + 9 espacios. Si el número de cifras de la parte entera del número que queremos imprimir es mayor que m, se imprimen todas estas cifras y se desplaza a la derecha el punto decimal

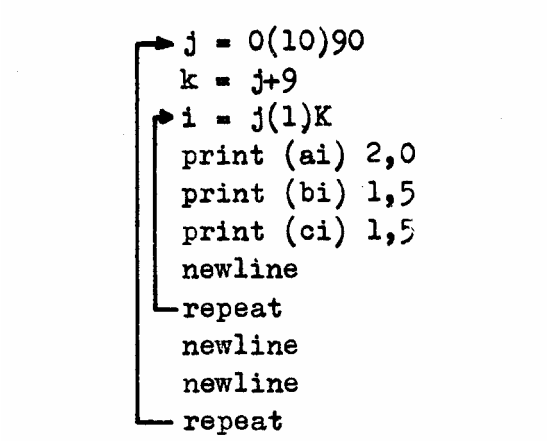
?

c) Impresiones parciales

El signo ? colocado al principio o final de una sentencia arit

mética, produce la impresión del número calculado en ésta, sobre la parte izquierda de la hoja del teletipo si una llave de la consola está en una posición determinada durante la traducción. Si el número es mayor que 10^{14} se imprimirá en forma de punto decimal flotante, en los demás casos se imprime con diez decimales. Los índices se imprimen como enteros sin punto ni parte decimal. No es lícito usar esta instrucción con órdenes de lecturas. Esta instrucción se utiliza especialmente durante la prueba de un programa.

Ejemplos: Imprimir ai, bi y ci (i = 0 (1) 99) en bloques de diez, separados por una doble línea en blanco. Las ai son enteros de dos cifras, las bi y ci deben darse con una cifra entera y cinco decimales.



5. Directivas. Hay sentencias en el sistema autocode que no tienen efecto durante el proceso de cálculo y que sólo sirven para organizar nuestro programa durante la traducción, para ubicar los distintos capítulos o para otras facilidades como son escribir el título de nuestro programa, o imprimir en un momento determinado el número de casillas disponibles en la memoria rápida. Ya conocemos una de las directivas de mayor uso representada por → , utilizada para reservar la zona de memoria correspondiente a las variables principales; a continuación veremos las directivas: title, chapter, close, variables, p s a.

title

a) Titulo

La directiva title tiene por efecto imprimir en el papel de la teletipo las palabras, números y signos que estén escritos a continuación en la próxima línea. Principalmente es usada para escribir el titulo de nuestro programa a la cabeza de los resultados, o algunos comentarlos sobre los mismos. Si el titulo ocupara más de una línea hemos de preceder cada línea del mismo con la palabra title.

Ejemplo: si escribimos en el programa

```
title
tabulación funciones de coulomb
title
delta = 0.0l    epsilon = 0.0l
```

aparecerá en la cabecera de nuestra tabla

```
tabulación funciones de coulomb
delta = 0.0l   epsilon = 0.0l
```

chapter close

b) Capitular

El par inseparable de directivas chapter α, y close, se emplea para indicar el principio y el fin de cada uno de los capítulos en que hemos dividido nuestro programa en caso de ser demasiado largo. Esto nos facilita la ubicación de cada capitulo en la memoria grande de la siguiente manera: la máquina cuando lee chapter α, reserva cierta zona a la memoria grande, cuando lee close ordena trasladar el capitulo actual a la zona reservada. El número α debe ser un entero comprendido en el rango (0 , 22).

En el caso especial del capitulo 0, la directiva close no sólo ordena trasladar el capitulo actual a la memoria grande sino que ordena comenzar el proceso de cálculo en su primera instrucción, por eso este capitulo debe ser leído el último.

_ w

c) Reserva de variables

La directiva → reserva zonas en la memoria actual de números donde se ubicarán las variables principales. Su uso ha sido ya explicado en II–l.l.–a.

variables

d) Conservación de la reserva

Si un capitulo opera con las mismas variables que el capitulo α, anterior en la cinta, no es preciso reservar de nuevo la zona para las variables, pues basta escribir debajo de la directiva del capitulo la directiva variables α, pudiendo utilizar además alguna variable nueva que se precise y que no estuviera contenida en dicho capítulo. Estas nuevas variables se reservan de la forma ordinaria mediante la flecha, después de haber escrito variables α.

p s a

e) Espacio disponible

Dado lo variable de la longitud de las sentencias no es posible saber de antemano el número de ellas que caben en un capitulo. Pero si queremos conocer con exactitud el número de registros

que quedan disponibles podemos utilizar la directiva p s a. Esta directiva insertada en cualquier parte del programa tiene por efecto imprimir el número del capitulo y el número de registros que quedan aún disponibles. El número máximo de instrucciones de máquina utilizables por capítulos es de 832.

f) Quickies

```
ψ sq rt
ψ cos
ψ log
ψ tan
ψ radius
ψ sin
ψ exp
ψ arctan
```

Los anteriores nombres de funciones incluidos inmediatamente antes de la directiva close**,** son considerados como directivas (llamadas quickies) que tienen por efecto incluir durante el proceso de traducción las rutinas para el cálculo de dichas funciones, si hubiere suficiente espacio disponible en el capitulo.
De esta forma nos evitamos que durante el proceso de cálculo tengamos que transferir dicha subrutina desde la memoria grande a la actual cada vez que aparezca una función. La lista de los quickies debe ser hecha en orden, poniendo los más frecuentes al principio. Si un quickie no es transferido al capitulo durante la traducción por demasiado largo, la máquina probará si el siguiente cabe para incluirlo. El número de instrucciones empleadas en cada quickie es el siguiente:
ψ sq rt - 48; ψ cos - 36; ψ log - 42; ψ tan - 42; ψ radius - 48; ψ sin - 36; ψ exp - 50; ψ arctan - 58. Los quickies ψ sin y ψ cos llaman a la misma subrutina por lo tanto en la lista sólo debemos incluir uno de ellos; lo mismo ocurre con los quickies ψ sq rt y ψ radius.
Ejemplo: En el siguiente esquema general de un capitulo vemos la forma y posición que ocupan los quickies

```
chapter 2
variables 1
v → 20
x → 5
instrucciones
ψ sq rt
ψ arctan
close
```

IV. MEMORIAS Y PROGRAMACION

Hasta ahora hemos dedicado nuestra atención a definir las distintas partes de la memoria actual de números (variables actuales, índices) así como a las partes de que está constituida una sentencia. A partir de este momento nos interesa la organización general de los programas y de las facilidades de que está provisto el lenguaje autocode para hacer más parecido el lenguaje matemático al lenguaje utilizado por dicho sistema. Con este fin estudiaremos previamente la disposición funcional de la memoria grande y la organización de la memoria actual de instrucciones, y atenderemos después a las facilidades y programas a que hemos hecho alusión anteriormente.

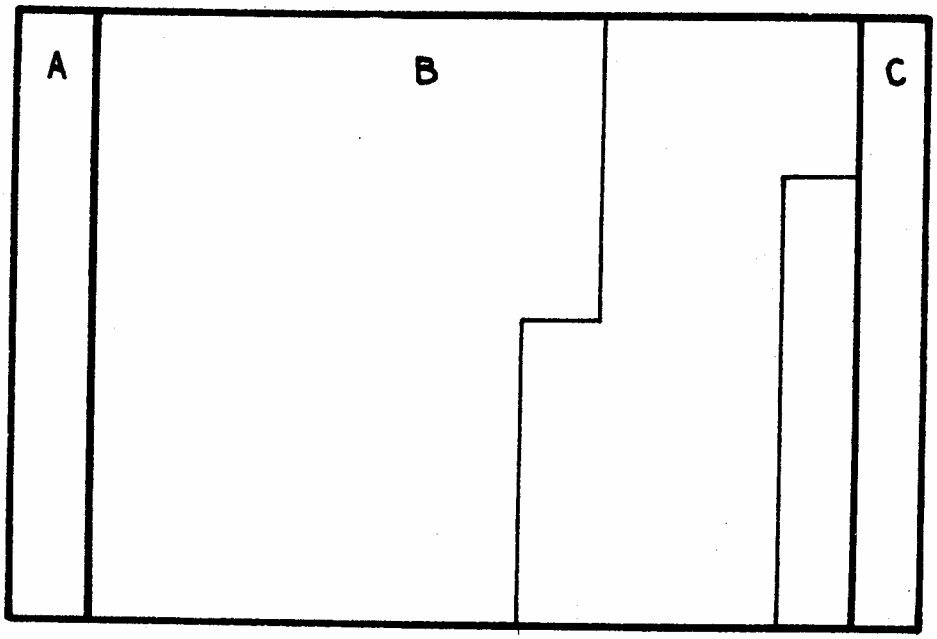


Fig.3.— (Véase con más detalle en Apéndice 1)

1. Memoria actual de instrucciones. La memoria actual de instrucciones consta de tres partes en las que se alojan las distintas partes del programa. En la primera están incluidas algunas secuencias como la división, cambio de capítulo,.. algunas constantes y espacio disponible para trabajo de las subrutinas. En la segunda (que es la más extensa) están contenidas las instrucciones y constantes que integran nuestro programa; si hubiese espacio se incluirán también en esta zona las subrutinas llamadas por los quickies. En la tercera parte se almacenar rá, cada vez que sea requerida, una de las funciones utilizadas por el sistema autocode (ψ sq rt, ψ sin, ψ exp.,...,).(Véase fig.3).

2. Memoria grande. La memoria grande está constituida por un tambor magnético, con capacidad para 13.824-casillas, a cada una de las cuales se le asocia un

número como dirección. Estas casillas están divididas en dos grandes grupos, según sean positivos o negativos los números que les corresponden como dirección; cada grupo está dividido en varias zonas con cometidos diferentes. Al primero de los grupos le corresponden casillas con direcciones positivas que van desde 0 a +10751; contienen las variables auxiliares y nuestro programa autocodificado completo, ubicado éste en las casillas de numeración más alta (10751, 10750,...); como cada capitulo ocupa 512 de estas casillas, la parte disponible para las variables auxiliares será de 10752−512n, siendo n el número de capítulos. A las casillas del segundo grupo se le asocian direcciones negativas comprendidas entre −1 y −3072 y consta de las siguientes zonas: una zona para el programa traductor autocode (−1 a −1536); tres zonas de almacenamiento temporario llamadas depósito de subcapítulo (−1537 a −2048), depósito de capitulo maestro (−2049 a −2560) y depósito especial (−2561 a −3072). La utilización de estas zonas la veremos al tratar de los programas. En caso de necesidad también el grupo de casillas con direcciones negativas puede usarse para alojar variables auxiliares. (Véase Apéndice 1).

3. Variables auxiliares. Cuando no son suficientes las 509 variables actuales podemos utilizar la zona de la memoria grande de direcciones positivas no utilizada por las instrucciones del programa como variables auxiliares; estas variables serán distinguidas por el número de su dirección. Como para ser procesados todos los números han de pasar por la memoria actual hemos de disponer de unas funciones que nos permitan las transferencias entre la memoria actual y la grande. Estas funciones son las siguientes: ψ 6 (α) v, n; ψ 7 (α) v, n; preserve y restore.

Indicamos con α una variable, un índice, una constante, o una expresión algebraica entre ellos. Siempre toma α un valor entero, truncando la parte decimal si la hubiere. Generalmente es usado en el lugar de α las variables especiales con acento. Indicamos con v una variable general o especial. Por último n indica a un índice o un entero.

ψ 6 (α) v, n

La función ψ 6 (α) v, n tiene por efecto transferir de la memoria grande a la actual el contenido de las n casillas que empiezan en la dirección α, a las variables actuales cuya locación inicial es v.

Ejemplos:

La instrucción

ψ 6 (10230) v_0, 5

transfiere el contenido de las posiciones 10230, 10231, 10232, 10233, 10234, a las variables v_0, v_1, v_2, v_3, v_4.

Son posibles expresiones como las siguientes,

ψ 6 (v_i+2v_j)w_i, h
ψ 6 (n) a, 1
ψ 6 (a') a_0 ,16

ψ 7 (α) v, n

La función ψ 7 (α) v, n tiene por efecto transferir de la memoria actual a la memoria grande, las n variables actuales cuya locación inicial es v, a las n variables auxiliares situadas en las direcciones de α a α + n – 1.

Ejemplo: la instrucción

ψ 7 (10230) v_0 ,5

transfiere los contenidos de v_0, v_1, v_2, v_3, v_4 a las posiciones de la memoria grande 10230, 10231, 10232, 10233, 10234.

preserve
restore

La instrucción preserve tiene por efecto almacenar el contenido de toda la memoria actual de números en determinada zona de la memoria grande. Su instrucción inseparable restore devuelve a la memoria actual lo que temporalmente se había almacenado en determinada zona de la memoria grande. El principal uso de este par de instrucciones es el empleo de una instrucción down entre ambas.

El subcapítulo llamado mediante un down puede a su vez llamar a un sub–subcapítulo, mediante otro down y por lo tanto se puede usar también el par preserve–restore, pero no así en el sub–subcapítulo. La zona de la memoria grande utilizada para el almacenaje temporal ordenado por el preserve del capitulo principal es el depósito de capitulo maestro, y los subcapítulos utilizan para el mismo objeto el depósito de subcapítulo. Hemos de notar que la variable especial π después de cada una de las instrucciones preserve, restore, down, up y across contiene la constante 3.141592...

V. LAS FACILIDADES

Para lograr mayor parecido con el lenguaje matemático, el sistema autocode está provisto de ciertas notaciones y funciones, que llamaremos facilidades, que son realmente instrucciones especiales de mucha más potencia que las vistas hasta aquí. Estudiaremos cuatro tipos de facilidades:

1. Aritmética de doble precisión
2. Álgebra compleja
3. Álgebra de matrices
4. Integración de sistemas de ecuaciones diferenciales.

1. Aritmética de doble precisión En las sentencias aritméticas ordinarias trabajábamos siempre con números que tienen como máximo nueve cifras significativas, bien sean consideradas en punto fijo o flotante. Algunas veces nuestros cálculos requieren que obtengamos los resultados con más de nueve cifras significativas y para ello hay que emplear las técnicas llamadas de doble precisión, que esencialmente consisten en considerar como un solo número las cifras contenidas en dos casillas diferentes. Por ejemplo, si queremos expresar con doble precisión el número 0.1237659380857364 almacenaremos en una casilla la parte más significativa, es decir 0.12376593 y la menos significativa en otra casilla, es decir 80857364. Si en general, dado un número M llamamos m_l a su parte más significativa y m_2 a su parte menos significativa, todo número M lo podemos representar por el par (m_l, m_2), y las operaciones aritméticas a realizar entre números de doble precisión serían las siguientes:

Suma $M(m_l,m_2)+N(n_l,n_2) = S(s_1,s_2)$
donde:
$s_1 = (m_l+n_l)_l$
$s_1 = (m_2+n_2)_1+(m_1+n_1)_2$

Producto $M(m_l,m_2)\ N(n_l,n_2) = P(p_1, p_2)$
donde:
$p_1 = (m_l\ n_l)_l$
$p_2 = (m_1n_2)_1+(m_2n_l)_1+ (m_1n_1)_2$

Cociente $M(m_1, m_2)/N(n_1, n_2) = Q(q_1, q_2)$

donde:

$$q_1 = (m_1/n_l)_1$$

$$q_2 = ((m_2 n_1 - m_1 n_2)/n_1^2)_1 + (m_l/n_l)_2$$

En las relaciones anteriores, el subíndice exterior al paréntesis expresa, según sea 1 o 2 la parte más o menos significativa de la operación exacta indicada en el interior del paréntesis. Hemos de notar que en todas las operaciones anteriores puede haber a veces un transporte de la parte menos significativa a la parte más significativa. También es evidente que si en la parte más significativa se cometen errores de redondeo es inútil aplicar la técnica de doble precisión. El sistema autocode tiene la facilidad de expresar mediante una sola instrucción cada una de las operaciones elementales utilizando la técnica de doble precisión; para ello empleamos las siguientes notaciones:

```
((A,B))=((X,Y))
((A,B))=((X,Y))+((U,V))
((A,B))=((X,Y))-((U,V))
((A,B))=1/((X,Y))
((A,B))=((X,Y))*((U,V))
((A,B))=((X,Y))/((U,V))
```

Expresamos simbólicamente mediante un par de variables actuales, encerradas entre paréntesis dobles y separadas por una coma, ((a, b)), a un número de doble precisión donde su parte más significativa está almacenada en la variable actual a y la parte menos significativa en la b. El primer miembro indica dónde deben ser almacenados los resultados del segundo miembro y por tanto a y b pueden ser únicamente variables actuales en cambio x, u, y, v pueden ser todas variables o x, u pueden ser números enteros si su correspondiente y, v es cero.
La parte menos significativa es siempre positiva.
Las variables que expresan las dos partes no tienen porqué ser consecutivas.
En operaciones con doble precisión no tiene efecto el signo ?.

double length

Siempre que utilicemos en un capítulo alguna de las facilidades de doble precisión hemos de incluir entre las directivas iniciales del capitulo, la directiva double length, que tiene por efecto

incluir al comienzo del capítulo las subrutinas de doble precisión.
Ejemplo: En el siguiente esquema general de un capitulo vemos la ubicación de la directiva double length:

```
CHAPTER 4
VARIABLES 2
Z→16
DOUBLE LENGTH

INSTRUCCIONES

xLOG
xEXP
CLOSE
```

2. Álgebra compleja. Otras facilidades que incluye el sistema autocode, son las operaciones elementales y algunas funciones, entre los números complejos mediante una simple instrucción, como analizaremos a continuación.

(α , β)

Definimos un número complejo en sistema autocode, como un par de variables o constantes, dadas en un cierto orden, separadas por una coma y encerradas entre paréntesis. La primera parte α (una variable, o una constante) representa la parte real, y la segunda β (variable, o constante) la parte imaginaria.
Ejemplos :
(A, B) (3.241, 6.742) (X (I+1),X(I+2))
(AI,BI) (AI, –3.94) (AI,A(I+1))

El lenguaje autocode nos permite hacer entre complejos las siguientes operaciones:

```
(U,V)=(X,Y)
(U,V)=(X,Y)+(A,B)
(U,V)=(X,Y)-(A,B)
(U,V)=(X,Y)*(A,B)
(U,V)=(X,Y)/(A,B)
(U,V)=xSQRT(X,Y)
(U,V)=xLOG(X,Y)
(U,V)=xEXP(X,Y)
   U =xRADIUS(X,Y)
```

U>o
π >V>-π

Hemos de notar de nuevo la importancia de la ortografía. También es importante observar que para el producto es preciso usar el asterisco. En operaciones complejas el signo ? no tiene efecto. Las funciones complejas sq rt, log, exp, utilizan para su cálculo las funciones reales: sq rt; para exp, sin y cos; para log, arctan; respectivamente; por tanto cada vez que se utilice una función compleja debemos incluir sus funciones reales correspondientes como quickies. Ejemplo: Calcular la expresión

$$u+iv = \sqrt{\frac{e^{x+iy}(2x+iy)}{(x-iy)^3}}$$

Para ello procedemos de la siguiente manera:

```
£1=2X
£2=-Y
(£3,£4)=XEXP (X,Y)
(£5,£6)=(£3,£4)*(£1,Y)
(£7,£8)=(£5,£6)/(X,£2)
(£9,£10)=XSQRT (£7,£8)
(U,V)=(£9,£10)/(X,£2)
```

Al final del capitulo donde estuviese incluida esta secuencia deberían aparecer los quickies:

ψ exp
ψ sin
ψ sq rt

3. Álgebra de matrices. Las matrices son almacenadas como una única columna, utilizando para ello un grupo de variables auxiliares; la primera posición de una matriz A la indicamos por el contenido de una variable especial, generalmente a'. Para localizar en una matriz de orden (m, n) el elemento a_{ij} utilizamos la siguiente expresión:

$$a_{ij} = a'(n\,(i-1) + (j-1))$$

el segundo miembro expresa que a partir de la posición inicial indicada en a', hemos recorrido (i−1) bloques correspondientes a (i−1) filas de n elementos, y que en el bloque i (es decir en la fila i) hemos recorrido j−1 elementos, esto es, nos encontramos en el elemento a_{ij} .

En sistema autocode, esta búsqueda es automática, así como las principales operaciones entre matrices, incluyendo las de lectura y escritura, para lo cual está provisto de unas funciones que dependen de varios parámetros y cuya característica es un entero que varia desde 8 hasta 28 según sea el cometido que tenga que ejecutar. Los parámetros son:

- una variable especial o un entero, para indicar la ubicación en la memoria grande del primer elemento de las matrices que intervienen en la función. Generalmente se emplean variables especiales con acento.
- una variable especial o un entero, para indicar el número total de elementos, o el orden, según los casos.
- un índice o un entero, para indicar el número de cifras enteras o fraccionarias que queremos imprimir. (También se usa un índice o un entero para indicar el orden de una matriz en el caso particular de la división).
- una variable especial, para indicar un escalar.

  Generalmente se usa una variable especial sin acento.

A continuación veremos los grupos de funciones que ejecutan los siguientes cometidos:

- Imprimir una matriz en punto fijo o flotante.
- Leer una matriz o un vector
- Combinación lineal de dos matrices, incluyendo los casos particulares de una matriz diagonal o unitaria.
- Trasponer una matriz.
- Hallar el determinante de una matriz.
- Multiplicación y división de matrices.

ψ 8 (a', u, v, m, n)
ψ 9 (a', u, v, n)

Imprimir

La función ψ 8(a',u,v,m,n) tiene por efecto imprimir la matriz A, almacenada en la memoria grande a partir de la posición registrada en a', de la siguiente forma:

–cada fila de la matriz ocupa sobre el papel una columna de v elementos.

–después de los v elementos de una fila sigue una línea en blanco.

–después de la fila en blanco sigue la fila siguiente de la matriz encolumnada con la anterior.

–cada elemento es impreso en la forma de punto decimal fijo, con signo si fuese negativo, m cifras enteras y n decimales. Se escribirán en flotante cuando su valor sea mayor que 10^{16}, o sí m=0.

Ejemplo: Supongamos la matriz

$$A = \begin{vmatrix} 2{,}512 & 2{,}753 & -3{,}104 \\ 5{,}215 & -7{,}126 & 9{,}627 \end{vmatrix}$$

almacenada en las casillas de la memoria grande de direcciones l00, 10l, 102, 103, 104, 105, y supongamos en a' está almacenado en número 100; la función ψ 8 (a', 2, 3, 1, 2) imprimirá la matriz anterior en la siguiente forma:

2 ,51
2,75
−3,10

5,21
−7,13
9,63

El mismo efecto tendría la función ψ 8 (l00, 3, 2, 1, 2).

La función ψ 9 (a', u, v, n), tiene el mismo efecto anterior con la salvedad de que imprime los números en forma de punto decimal flotante, con n = l0 cifras en la mantisa y con exponente entero menor que 70. Tiene exactamente el mismo efecto que la función ψ 8 con m = 0.

ψ 10(a',w)

Leer

Esta función tiene por efecto leer los w números perforados sobre la cinta en forma de punto fijo o flotante y almacenarlos en las casillas de la memoria grande cuya primera ubicación está registrada en la variable especial a'. Si w=u.v puede considerarse este conjunto de números como una matriz de orden (u, v). En cualquier caso puede considerarse como un vector de w dimensiones.

Combinación lineal. A continuación damos las funciones que nos permiten calcular la combinación lineal de dos matrices; por tener algunas particularidades diferentes las dividiremos en tres grupos, según se trate de la combinación lineal de dos matrices cualesquiera, de una matriz cualesquiera y de la unitaria, o de una cualesquiera y otra diagonal. Naturalmente en los dos últimos casos las matrices han de ser cuadradas.

− dos matrices cualesquiera

a'= ψ 11(b', c',w)	A = B + C	orden(u, v)
a'= ψ 12(b', c',w)	A = B−C	orden(u, v)
a'= ψ 13(b', x, c',w)	A = B + x C	orden(u, v)
a'= ψ 14(b', x, c',w)	A = B – x C	orden(u, v)
a'= ψ 15(b',w)	A = B	orden (u, v)

Las variables especiales a', b', c', contienen las direcciones de la primera casilla de las zonas que van a ocupar respectivamente las matrices A, B, C. La variable especial x indica un escalar. Las tres matrices han de ser del mismo orden y w contiene su número de elementos, es decir w = u . v.

Ejemplos: Supongamos las matrices

$$B = \begin{vmatrix} 1 & 2 & 2 \\ 2 & 1 & 1 \end{vmatrix} \qquad C = \begin{vmatrix} 2 & 1 & 1 \\ 1 & 2 & 2 \end{vmatrix}$$

almacenadas respectivamente en las casillas de memoria grande numeradas 20, 21, 22, 23, 24, 25, y 26, 27, 28, 29, 30, 31. Si las variables especiales b', c', a' contienen respectivamente los números 20, 26, 32, la función

$$a' = \psi\ 12\ (b', c', 6)$$

almacenará en las casillas 32, 33, 34, 35, 36 la matriz:

$$A = \begin{vmatrix} -1 & 1 & 1 \\ 1 & -1 & -1 \end{vmatrix}$$

Supuestas las mismas condiciones anteriores y además 0,5 almacenado en a, entonces la función

$$a' = \psi\ 13\ (b', a, c', u)$$

tiene por efecto almacenar la matriz

$$A = \begin{vmatrix} 2 & 2.5 & 2.5 \\ 2.5 & 2 & 2 \end{vmatrix}$$

en las casillas 32, 33, 34, 35, 36, 37.

– de una matriz cualesquiera y la matriz unidad.

$a' = \psi\ 17(b', u)$	$A = B + I$	orden (u, u)
$a' = \psi\ 18(b', u)$	$A = B - I$	orden (u, u)
$a' = \psi\ 19(b', x, u)$	$A = B + x\ I$	orden (u, u)
$a' = \psi\ 20(b', x, u)$	$A = B - x\ I$	orden (u, u)

En este caso las matrices son cuadradas, y u representa el orden, no la totalidad de elementos como en el caso anterior.

– una matriz cualesquiera y otra diagonal.

$a' = \psi\ 21\ (b', d', u)$	$A = B + D$
$a' = \psi\ 22\ (b', d', u)$	$A = B - D$
$a' = \psi\ 23\ (b', x, d', u)$	$A = B + x\ D$
$a' = \psi\ 24\ (b', x, d', u)$	$A = B - x\ D$

También aquí las matrices son cuadradas de orden u. La matriz diagonal se almacena como un vector de dimensión u, en una zona de la memoria grande cuya primera dirección está almacenada en d'.

a' = ψ16 (b', u, v)

Traspuesta

Esta función tiene por efecto almacenar en las casillas indicadas por a', la matriz traspuesta de la b', es decir que si la primera es de Orden (u, v) la segunda es (v, u).

x = ψ25 (a', u)

Determinante

Hace aparecer en la variable x el valor del determinante de la matriz de orden u almacenado en la zona indicada por a'.

Multiplicación y división

a'= ψ 26 (b', c', u, v, w) A (u, v) = B (u,w) C (w,v)
a'= ψ 27 (b', c', u, v, w) A (u, v) = B (u,w) C'(w,v)
a'= ψ 28 (b', m, n) A (m, n) = B^{-1}(m,m) A (m,n)

Expresamos por C' a la traspuesta de C. Hemos de notar que aquí indicamos por m, n el orden de las matrices en lugar de u, v como hacíamos anteriormente, ya que hemos de utilizar índices o enteros menores que 511 para este fin. También conviene saber que la división destruye la matriz B.

Ejemplo: Si queremos hacer un programa que nos permita hallar las soluciones de un sistema de ecuaciones lineales

$$a_i x_i = b_i$$

de orden $n \leq 20$, e imprimir dichas soluciones encolumnadas con r cifras enteras y s decimales, procederíamos de la siguiente forma:

```
CHAPTER o
X→20

READ (N)
READ (R)
READ (S)
Q=NN
An=o
Bn=Q
XIo (An,Q)
XIo (Bn,N)
Bn=X28 (An,N,N)
X6 (Bn) XI,N
I=I(I)N
PRINT (XI) R,S
NEW LINE
REPEAT
END

CLOSE
→
```

4.- Integración de ecuaciones diferenciales. El sistema autocode está provisto de una facilidad que nos permite integrar un sistema de ecuaciones diferenciales .por el método de Runge—Kutta, mediante una sola instrucción (int step (m)), si previamente hemos reservado las variables que vamos a utilizar, determinado los parámetros que intervienen (número de ecuaciones, longitud del paso, condiciones iniciales) y definido las ecuaciones, como especificamos a continua

int step (m)

Tiene por efecto integrar el sistema de ecuaciones diferenciales

$$f_i = dy_i / dx = f_i (x, y_1, y_2,...y_n)$$

donde i = (l, 2, 3,...n). Previamente hemos de reservar las variables principales f_i, y_i, g_i, h_i (las dos últimas como espacio de trabajo); determinar los valores de los índices n (número de ecuaciones) y h (longitud del paso); determinar cuáles son los valores iniciales de x, y_1, y_2,...y_n; y definir las ecuaciones entre la marca m y la instrucción 592,0.
Ejemplo: Integrar el sistema

$$f_1 = dy_1 / dx = (y_2 + y_3)^{½}\ e^{y1}$$
$$f_2 = dy_2 / dx = y_1^{½} \log_e x$$
$$f_3 = dy_3 / dx = 6.9 \cos (y_3 + y_4) + x$$
$$f_4 = dy_4 / dx = y_2 + y_3 \arctan (y_1 + y_5)$$
$$f_5 = dy_5 / dx = (y_2\, y_3\, y_4 + y_5\, y_1\, y_2) / x$$

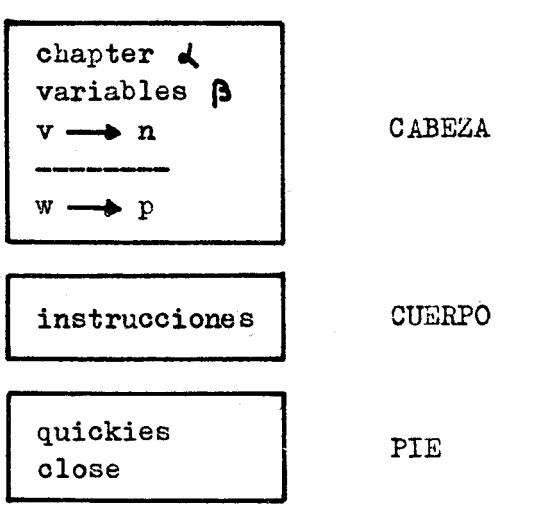
en el intervalo (10, 10.25) con un paso de 0.025, y para valores iniciales $y_1(10)=0$; $y_2(l0)=0.33$; $y_4(10)=2$; $y_5(10)=0$. Para integrar este sistema nuestro programa sería el siguiente:

```
CHAPTER o

F→5
G→5
H→5
Y→5

H=.025
N=5

X=10
Y1=0
Y2=1
Y3=.33
Y4=2
Y5=0

1)NEW LINE
PRINT (X) 1,3
I=1(1)5
PRINT (YI) 4,2
REPEAT

INT STEP (10)

JUMP 1,10.25>X
END

10)A=XSQRT (Y2Y2+Y3Y3)
B=XEXP (Y1)
F1=AB
A=XSQRT (Y1)
B=XLOG (X)
F2=AB
A=XCOS (Y3+Y4)
F3=6.9A+X
A=XARCTAN (1,Y1+Y5)
F4=Y2+AY3
F5=XDIVIDE (Y2Y3Y4+Y5Y1Y2,X)
592,0

XSQRT
XEXP
XLOG
XCOS
XARCTAN

CLOSE
```

VI. LOS PROGRAMAS

Un programa completo contiene varias partes y éstas a su vez están divididas en otras menores; al programa total lo llamaremos macroprogramas, que estará dividido en varios subprogramas, cada uno de los cuales constará de uno o más capítulos. Los distintos subprogramas de un macroprograma están enlazados por medio de un subprograma llamado programa maestro. A continuación veremos los convenios formales que deben verificar los capítulos, los subprogramas (o programas si no son referidos a un macroprograma) o la organización general de un macroprograma, cuando programamos en autocode.

1. Los capítulos. Los capítulos constan de tres partes esenciales: cabeza, cuerpo y pie. La cabeza y el pie están formados por directivas y sirven para organizar la traducción de nuestras instrucciones; el cuerpo constituye el conjunto de las instrucciones que utilizaremos para la ejecución del programa. El siguiente esquema muestra la estructura más general de un capitulo:



En algunos casos pueden omitirse las directivas variables, →, (quickies), pero siempre que se usen se deben hacer en el orden relativo indicado en el esquema. En todos los casos se precisan las directivas chapter α, y close.

Ejemplos:

a) Tabular la función de variable compleja

$$w = \text{sen } z$$

(donde $w = u + iv$ y $z = x + iy$) en los puntos del plano x, y situados .sobre la curva

$$y = x^2 + 1$$

con abscisa que varía en el intervalo (0,1) con un paso de 0.01.

```
CHAPTER 1
£→2
```

```
I=0(1)100
X=.11
Y=XX+1
£0=XSIN (X)
£1=XEXP (Y)
£2=.5/£1
U=.5£0£1+£0£2
£0=XCOS (X)
V=.5£0£1-£0£2
NEWLINE
PRINT (X) 1,1
PRINT(Y) 1,2
SPACE
PRINT (U) 1,3
PRINT (V) 1,3
REPEAT
END
```

```
xSIN
xEXP
CLOSE
```

b) Hallar por el método de Simpson el valor de $\int_{-1,1} e^x dx$ con un error

menor que D

```
READ (D)
H=2
F=1
G=0
N=1
A=XEXP (1)
E=A+1/A
B=E+4
B=B/3
1)N=2N
H=.5H
A=.5H-1
G=G+F
F=0
I=1(1)N
X=XEXP (A)
F=F+X
A=A+H
REPEAT
C=E+4F+2G
C=HC/6
Y=XMOD (B-C)
B=C
JUMP 1,Y≥D
PRINT (C)1,9
END
```

OBSERVACION. Nótese que los puntos pares de una partición coinciden con los pares e impares de la anterior, y por tanto solo precisamos los impares de cada nueva partición.

Recordamos de nuevo que el número de instrucciones de máquina máximo que puede ocupar un capítulo es de 832, para controlar el número de instrucciones utilizadas tenemos las directivas p s a (imprimir espacio disponible) y en caso de que fueran más de 832 acusaría el error (imprimiendo el número del capitulo y las instrucciones no contenidas en la memoria por falta de capacidad). Cada capítulo ocupa una zona fija en la memoria grande según sea su número. Las mismas marcas pueden ser utilizadas en capítulos diferentes. Siempre se debe marcar la primera instrucción de cada capítulo. Cada vez que cambiamos de capítulo se restaura en la variable especial el número *3.1415...* Siempre deben estar en un mismo capítulo extremos de ciclo i = p(q)r, repeat y las n) = 4) o n)=m), jump(n).

2. Los programas. Podemos definir un programa como uno o más capítulos capaces de efectuar una etapa determinada de nuestro proceso de cálculo. Hay dos tipos esencialmente distintos: los subprogramas y el programa maestro. Se dispone de un conjunto de subprogramas de gran generalidad elaborados por los usuarios del sistema autocode que constituyen la biblioteca.

2.1. Los subprogramas, tienen una misión de cálculo autónoma y en general pueden ser utilizados en macroprogramas muy diferentes. Sería un subprograma, por ejemplo, el conjunto de capítulos necesarios para calcular los autovectores y autovalores de una matriz dada. Para poder utilizar un programa necesitamos un nombre mediante el cual llamarlo, e indicarle dónde se encuentran los datos y dónde colocará los resultados; lo primero lo hacemos mediante el título y lo segundo y tercero mediante los parámetros.

programme – α

Titulo

Para distinguir un programa de otro y poderlo ensamblar formando un macroprograma, designamos a cada uno la directiva programme–α donde α debe ser un entero comprendido entre l–1023, adoptándose el convenio de reservar los números mayores de 500 para identificar los programas de la biblioteca. Es imprescindible el uso del guión entre programme y α.

Los parámetros

Los parámetros son variables o índices que deben tomar valores determinados de acuerdo con el caso al que apliquemos el programa. El "nombre" de la variable o índice está fijado por el programa, el valor numérico almacenado en dicha variable o índice depende de la aplicación concreta que hacemos del programa. Con este fin utilizaremos las variables especiales siguiendo el siguiente convenio:

1.– Los argumentos individuales, las dimensiones de zona, etc. ... serán expresados mediante variables especiales sin acento, o índices.

2.− Las zonas de números (vectores, matrices...) serán indicados mediante una variable especial con acento que contendrá la dirección del primer elemento de la zona.

3.− Los resultados se expresarán de forma análoga, según sean escalares o vectoriales.
Hemos de notar que conviene utilizar la directiva preserve antes de dar los valores concretos a nuestros parámetros, y restore inmediatamente después de la instrucción mediante la que entramos en el subprograma, de esta forma conservamos el contenido de la memoria actual e impedimos que durante la ejecución del subprograma se destruya su contenido.
Ejemplo: Supongamos que el programme−506 calcula las raíces reales y complejas del polinomio con coeficientes reales o complejos

$$\alpha_0 z^n + \alpha_1 z^{n-1} + \ldots + \alpha_{n-1} z + \alpha_n$$

Es evidente que antes de entrar en el subprograma necesitamos conocer
− el grado de la ecuación.
− la parte real de los coeficientes.
− la parte imaginaria de los coeficientes,
y para salir saber dónde están:
− la parte real de las raíces.
− la parte imaginaria de las raíces.
− la precisión de las raíces.

Si suponemos almacenado en n el grado de la ecuación, en a' la dirección de la parte real del primer coeficiente, en b' la dirección de la parte imaginaria del primer coeficiente, en e la precisión, y en c' y d' las direcciones de las partes real e imaginaria de las raíces, las variables e índices n, e, a', b', c', d' serán los parámetros del programme−506, y en un caso particular para una ecuación de sexto grado antes de llamar al subprograma necesitaríamos escribir

```
------
preserve
n = 6
e = 0.001
a' = 100
b' = 106
c' = 112
d' = 118
down 1/1−506
restore
------
------
```

donde los coeficientes estarán almacenados en las casillas 100−111 de la memoria grande y las raíces se alma cenaran en las 112−123 de la misma memoria.

.2.2. El programa maestro, tiene por finalidad ensamblar todos los subprogramas que van a intervenir en nuestro proceso de cálculo, es decir formará los parámetros y ordenará los saltos. Además de las operaciones de ensamblaje ejecutará aquéllas que por su brevedad o sencillez no necesiten un programa especial.
El programa maestro en general no necesitará un titulo, precisándolo sólo cuando entremos a un subprograma mediante una instrucción across.

2.3. La biblioteca. El Computing Machine Laboratory de la Universidad de Manchester, en colaboración con la casa Ferranti y todos los usuarios de la máquina Mercury (*) han elaborado una biblioteca que contiene programas de gran generalidad útiles para ser empleados en procesos de cálculo de tipo muy distinto. Cualquiera de estos programas los podemos incorporar a nuestro macroprograma con las mismas condiciones con que incluíamos cualquier otro subprograma. Algunos de los programas de la biblioteca no son tan completos, como por ejemplo ocurre con el programa para integrar una función, que necesitamos agregarle la rutina que genere el interesado. Es evidente que para el empleo de programas de la biblioteca necesitamos conocer sus especificaciones, es decir los parámetros que usan las variables o índices que necesita como hemos de hacer la entrada.
En la página siguiente incluimos el pliego de especificaciones que acompaña el programa para el cálculo de las raíces de un polinomio.

(*) Computing Machine Laboratory—Manchester University
Computing Center. Oxford University
London University Computing Group
United Kingdom Atomic Energy Authority
United Kingdom Atomic Energy Risley (Industrial Group)
United Kingdom Atomic Energy Harwell
United Winfrith Heath
Royal Aircraft Establishment. Farmboroug
Meteorological Office
Imperial Chemical Industries Ltd.
Associated Electrical Industries Ltd. (AEI)
General Electrical Company Ltd. (GEC)
Shell Petroleum Ltd.
BP Petroleum Ltd.
Comisión Francesa de Energía Atómica. Saclay
Comisión Belga de Energía Atómica
Comisión Sueca de Energía Atómica
Instituto Noruego de Investigaciones de la Defensa
CERN. Ginebra
Instituto de Cálculo. Universidad de Buenos Aires.

A -506

Título: Programme-506 Solución de ecuaciones algebraicas.
Objeto: Este programa calcula los ceros del polinomio

$$a_o z^n + a_1 z^{n-1} + \ldots + a_n = 0$$

donde las a_i pueden ser reales o complejas.

Descripción: El método usado está descripto por D.E. Muller en M.T.A.C., Oct. 1956 (p. 208).

Cada raíz es determinada por un método iterativo seguido de la eliminación del correspondiente factor de la ecuación. La precisión relativa requerida en las raíces está especificada por un parámetro de programa e el cual es empleado como un criterio de convergencia, así

$$\left| \frac{x^{(m+1)} - x^{(m)}}{\frac{1}{2}x^{(m+1)} + \frac{1}{2}x^{(m)}} \right| < e$$

donde $x^{(m)}$, $x^{(m+1)}$ son dos sucesivas iteraciones de una raíz x.
Así, para obtener cuatro cifras significativas, poner e=0.001

Parámetros:

n	orden de la ecuación (≤ 116)
e	precisión relativa de las raíces
a'	las partes reales de las a_i están almacenadas en a', a'+1,...a'+n.
b'	las partes imaginarias de las a_i están almacenadas en b', b'+1,... b'+n.
c'	las partes reales de las raíces están almacenadas en c', c'+1,... c'+n-1.
d'	las partes imaginarias de las raíces están almacenadas en d', d'+1,... d'+n-1.

Entrada: El programa consiste en un único capítulo llamado como sigue:

1. Si las a_i son complejas por "down 1/1 - 506".
2. Si las a_i son reales por "down 2/1 - 506" (en este caso no es necesario especificar b').

Tiempo: Por ejemplo 1 min. para n=25, 4 min. para n=50 y 16 min. para n=100.
Precisión: Véase más arriba en Descripción.

Autor: R.H. Kerr de la Universidad de Manchester
Fecha: 27 de abril de 1959.

Para distinguir los programas de la biblioteca, de los elaborados por nosotros ante cada problema concreto, a aquéllos les asignamos un número comprendido entre 501 y 1023, mientras que a éstos les debemos asignar números menores que 501. A continuación damos una lista de los programas existentes actualmente en la biblioteca, aunque como es natural, su con tenido es cambiante, bien incrementándose en número, o bien mejorando los ya existentes; en la siguiente lista damos el título del programa y el objeto del mismo:

Programme−50l Cálculo del límite de una sucesión numérica.
Dados los (n + l) términos de una sucesión numérica

$$a_0 \quad a_1 \quad a_2 \quad ... \quad a_n$$

este programa calcula su limite para $n \to \infty$

Programme−502 Simple cuadratura.
Este programa calcula el valor de la integral

$$\int_{a,\,b} f(x)\,dx$$

donde f(x) carece de singularidades en el intervalo real (a, b).

Programme−503 Cuadratura
Este programa calcula

$$\int_{a,\,b} (b-x)^u\,(x-a)^v\,f(x)\,dx$$

donde u> –l, v> –l y f(x) no tiene singularidades en el intervalo real (a, b).

Programme−504 Cuadratura de integrales infinitas.
Este programa calcula el valor de la integral

$$\int_{c,\,\infty} f(x)\,dx$$

donde f(x) carece de singularidades en el intervalo real (c,oo)

Programme−505 Análisis armónico.
Dados los valores de una función periódica f(x) = f(x + L) en los 2n + l puntos

separados igualmente $x_r = x_0+(r\ L/2n)$, r = 0 (l) 2n, cubriendo el periodo L, el programa calcula los coeficientes de la aproximación armónica.

$$f(x) = a_0 + \sum_{1}^{n} \left\{ a_r \cos \frac{2\pi r}{L} (x-x_0) + b_r \operatorname{sen} \frac{2\pi r}{L} (x-x_0) \right\}$$

Programme–506 Solución de ecuaciones algebraicas.
Este programa calcula los ceros del polinomio

$$a_0 z^n + a_1 z^{n-1} + \ldots\ldots + a_n = 0$$

donde las ai pueden ser reales o complejas.

Programme–507 Autocorrelación y correlación cruzada.
Dadas dos sucesiones (que pueden ser idénticas)

$$x_0\ x_1 \ldots x_n \qquad y_0\ y_1 \ldots\ldots y_n$$

el programa calcula las cantidades

$$Z_s = \frac{A_s - \frac{B_s - D_s}{n-s}}{\sqrt{\left(C_s - \frac{B_s^2}{n-s}\right)\left(E_s - \frac{D_s^2}{n-s}\right)}} \qquad \text{para } S = 0(1)\ p$$

$$A_s = \sum_{t=1}^{n-s} x_t y_{(t+s)} \ ; \ B_s = \sum_{t=1}^{n-s} x_t \ ; \ C_s = \sum_{t=1}^{n-s} x_t^2 \ ; \ D_s = \sum_{t=s+1}^{n} y_t \ ; \ E_s = \sum_{t=s+1}^{n} y_t^2$$

Programme–508 Resolver y tabular un sistema de ecuaciones diferenciales.

El programa resolverá y tabulará la solución del sistema de ecuaciones diferenciales

$$dy_i / dx\ f_i (y_1 , y_2 , y_3 \ldots . y_n ; x) \qquad i = 1 (l) n$$

en un intervalo uniforme d, comenzando con las condiciones iniciales $y_i (x_0)$ para $x = x_0$.

Programme–509 Tabulación

Este programa tabula en dos dimensiones la matriz a_{ij}, i=0(l)m, j=0(l)n. (Si no cupiesen todas las columnas en un ancho de papel, las columnas que exceden se escribirán debajo de las columnas anteriores).

Programme–510 Solución de ecuaciones normales de mínimos cuadrados.
Dado un sistema de m ecuaciones lineales de n incógnitas (m > n)

$$A\,x = b$$

donde A = (a ij), x = (xj), y b = (bí), i = l(l)m, j = l(l)n, este programa construye y resuelve las ecuaciones normales

$$A'Ax = A'b$$

Programme–511 Autovectores y autovalores de una matriz real general.

Este programa calcula los autovalores y autovectores de una matriz A almacenada en la memoria grande.

Programme—512 Entrada y salida de números con doble precisión.

Este programa lee números decimales de la cinta de datos y los almacena en pares de casillas consecutivas de la memoria grande. También este programa perfora números de doble precisión, almacenados en pares consecutivos de casillas de la memoria grande, con punto fijo redondeando la parte decimal.

Programme–515 Solución de un sistema de ecuaciones lineales con doble precisión.
Este programa resuelve un sistema de ecuaciones lineales

$$a_{ij}\,x_j = b_i \qquad i=l(1)n$$

donde las a_{ij} y b_i son dadas con doble precisión. Las ecuaciones son almacenadas por filas siendo ubicadas las b_i a continuación de a_{in}.

Programme–516 Tabular la solución de un sistema de ecuaciones diferenciales.
Este programa es una versión modificada del programme–508.

Programme–518 División de matrices.
Este programa ejecuta lo mismo que la función ψ 28, es decir verifica

$$A = B^{-1}A$$

la única ventaja es que no necesitamos leer todas las subrutinas de las facilidades de matrices.

Programme–519 Cuadratura impropia real.
Este programa calcula la integral impropia I, como el límite

$$I = \int_a^b f(x)dx = \lim_{n\to\infty} \sum_{r=0}^{n} \int_{x_r}^{x_{r+1}} f(x)dx$$

donde $x_n = 2^{-n}(a-b) + b$

Programme–521 Cuadratura impropia compleja.
Esta es una versión del Programme–519 que acepta valores complejos del integrando, límites complejos, y una singularidad compleja. La tolerancia se aplica al módulo del resultado.

Programe–522 Entrada y salida de textos
Este programa lee textos de una cinta, los almacena y los perfora en la forma requerida.

Programme–523 Polinomio de mínimos cuadrados.
Este capitulo ajusta por mínimos cuadrados un polinomio de grado k a n puntos, no necesariamente de igual peso o igualmente espaciados.

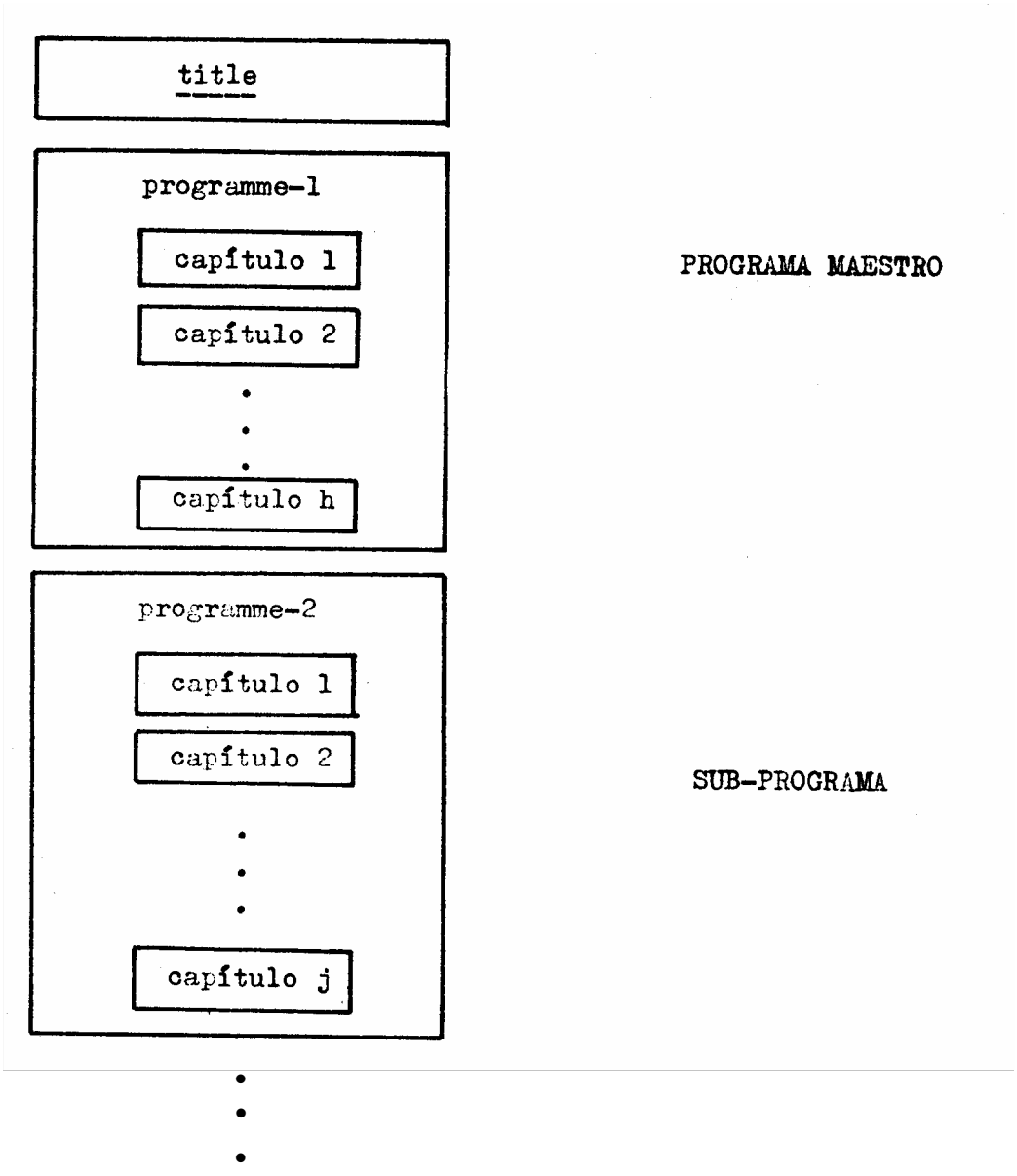
3. Macroprogramas. Como resumen de lo tratado hasta aquí, veremos finalmente la organización general de un proceso de cálculo que contenga varios subprogramas, es decir de un macroprograma. Para lo cual señalaremos que es conveniente realizarlo en los siguientes cuatro pasos;

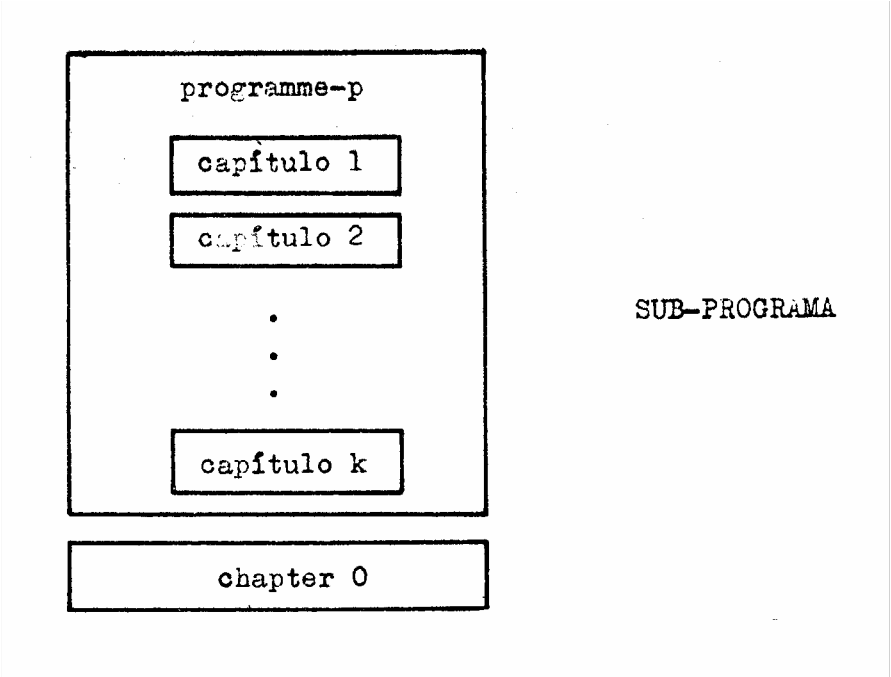
a – Hacer el diagrama en bloque, especificando las partes esenciales del proceso de cálculo que hemos de ejecutar.
b – Ver las partes que pueden ser tratadas utilizando la biblioteca de programas, y estudio de sus especificaciones.
c – Escribir los programas para las otras partes autónomas que carecen de programa adecuado en la biblioteca. Para ello conviene hacer un diagrama de detalle.
d – Escribir el programa de ensamblaje o programa maestro, teniendo presente cómo hemos de hacer la entrada de datos y cómo queremos que se efectúe la salida. Realmente el programa maestro responde al diagrama en bloque a que aludimos en a).

Hemos de hacer algunas salvedades referentes al empleo de algunas directivas y de las instrucciones de salto en los macroprogramas:

– la directiva variables puede sólo referirse a capítulos contenidos dentro de un mismo subprograma.
– las instrucciones de salto entre capítulos pertenecientes a subprogramas distintos deberán tener tres parámetros: la marca, el capítulo, el programa. Es decir que serán de la forma across M/C—P, down M/C—P.

La estructura funcional de un subprograma será en general la que indica el siguiente esquema:





A continuación damos un ejemplo de cómo utilizar un subprograma incompleto de biblioteca. Se trata de integrar mediante el programa A–502 la función $y = sen^2 x$ en el intervalo $(0, \pi/2)$. Para lo cual necesitamos indicar la función que queremos integrar al final del programme–502 que quedó incompleto con tal fin. Y antes de entrar necesitamos definir los parámetros de dicho programa; según las especificaciones debemos colocar en A y B los limites de integración, en E la precisión pedida o lograda, en u' la dirección de la memoria grande en donde queremos que se deposite el valor de la integral. La entrada se hace mediante la instrucción down 1/1–502. Como en este caso hemos supuesto que nuestro objeto es únicamente saber el valor de la integral incluimos unas instrucciones de salida, pero en problemas más generales el valor de la integral será un dato intermedio y por lo tanto no será preciso su impresión. Las siguientes instrucciones muestran esquemáticamente cómo procederíamos:

```
TITLE
INTEGRAL SENO CUADRADO


PROGRAMME - 502
CHAPTER 1

1)----------
----------
----------
----------
101)---------
----------
----------
----------
----------
UP
----------
----------
120)570,0
X=XSIN(X)
X=XX
JUMP101


CLOSE




CHAPTER 0
X>1


A=0
B=£/2
E=.0001
Un=0
DOWN 1/1-502
X6 (Un) X0,2
PRINT (X0) 2,4
PRINT (X1) 0,4


CLOSE
```

APEDICE I

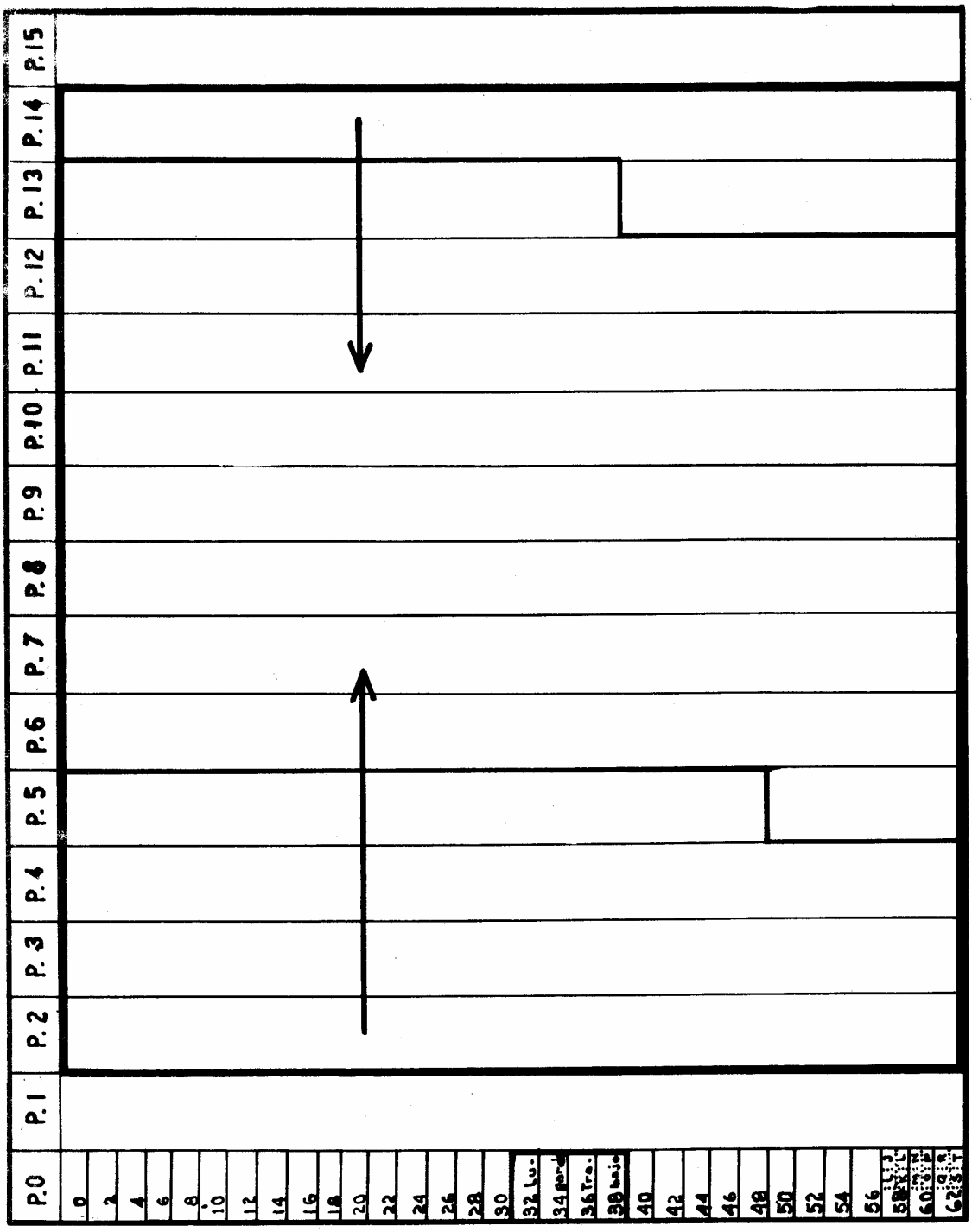
NOTAS SOBRE LAS MEMORIAS

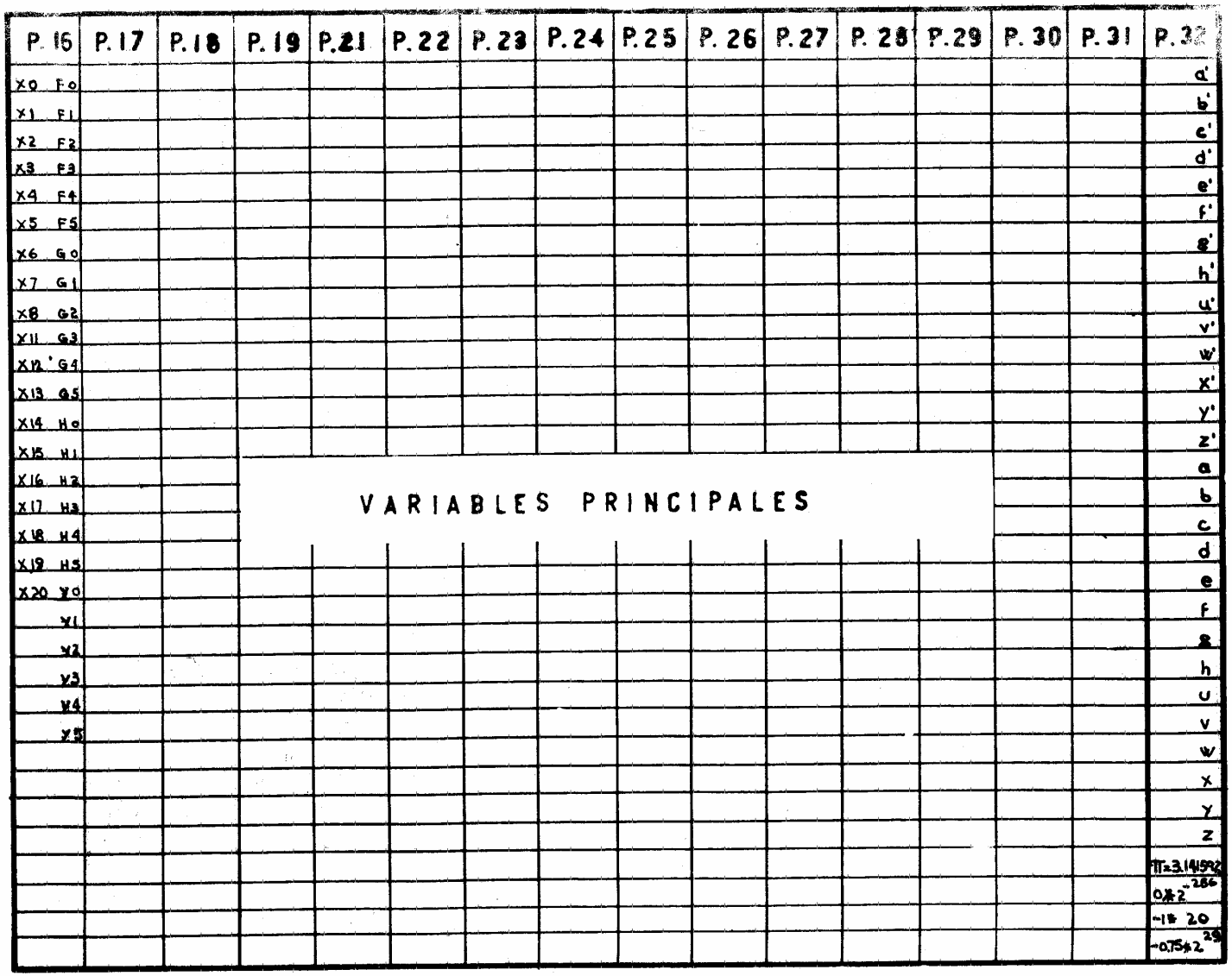
I. LA MEMORIA RAPIDA.

Ya vimos anteriormente que la memoria actual o rápida la podíamos considerar en dos partes diferentes, una para instrucciones y otra para números; vamos a ver ahora con detalle, cómo están divididas materialmente.

La memoria de instrucciones está dividida en 16 partes iguales llamadas páginas y numeradas del 0 al 15. Cada una de las página puede considerarse compuesta de 32 registros con capacidad para un número de 10 cifras decimales (40 dígitos binarios), numeradas con los números pares del 0 al 62, o de 64 registros con capacidad para 20 dígitos binarios, numeradas de una en una desde 0 a 63, o compuesta de 128 registros de 10 dígitos binarios numeradas sucesiva mente por los símbolos 0, 0+, 1, 1+,... 63, 63+. Visto esto, podemos decir que los índices ocupan los registros 58, 58+, 59, 59+, 60, 60+, 61, 61+, 62, 62+, 63, 63., de la página 0, como podemos ver en la figura 4. También en la misma figura observamos que en los registros 4 a 31, está siempre la rutina de división y los registros 32, 34, 36, 38, los podemos utilizar como lugar de trabajo (véase utilización de instrucciones de convencional en autocode). La página 1 esta reservada para transferencias especiales para ejecutar la raíz cuadrada compleja La página 15 está reservada para albergar una función, que no haya sido incluida como quickie. En las páginas 2 a 14 estará almacenada la información traducida correspondiente a uno de nuestros capítulos de la siguiente forma: a partir del registro 0 de la página 2 en adelante, las instrucciones; a partir del registro 62 de la página 14 hacia atrás las constantes, que figuran explícitamente en nuestro capitulo. En el espacio restante, entre la última instrucción y la última constante, las funciones incluidas como quickies, si hubiese suficiente espacio.

La memoria de números está compuesta por 16 páginas numeradas del 16 al 31. Las variables especiales tienen ubicación fija en la página 31 correspondiéndole los registros 0, 2, 4, ... 54 a las variables a'b'... z, la variable π que contiene, mientras no se modifique, la constante 3.141592... está en el registro 56 y en los registros 58, 60, 62 están almacenadas las constantes 0, −1, -4.10^8. Conviene notar que cuando aparece una instrucción preserve los índices pasan a ocupar los registros 58, 58+, ... 63+ de la página 31 antes de trasladar todo el contenido de la memoria de números a la memoria grande. Las páginas 16 a 30 contienen las variables especiales en el orden que hayamos dispuesto mediante la directiva $\rightarrow$ previamente a las instrucciones de cada capítulo. En la figura 5 representamos a la izquierda de los registros de la página 16, las ubicaciones que le corresponderían a las variables del ejemplo de la página 33, y a la derecha observamos que los mismos lugares les corresponden a las primeras veinte variables reservadas en el ejemplo de la página 35.


P.15
P.14
P.13
P.12
P.11
P.10
P.9
P.8
P.7
P.6
P.5
P.4
P.3
P.2
P.1
P.0
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
34
36
38
40
42
44
46
48
50
52
54
56
58
60
62


P. 16
P. 17
P. 18
P. 19
P. 21
P. 22
P. 23
P. 24
P. 25
P. 26
P. 27
P. 28
P. 29
P. 30
P. 31
P. 32
X0 F0
X1 F1
X2 F2
X3 F3
X4 F4
X5 F5
X6 G0
X7 G1
X8 G2
X11 G3
X12 G4
X13 G5
X14 H0
X15 H1
X16 H2
X17 H3
X18 H4
X19 H5
X20 Y0
Y1
Y2
Y3
Y4
Y5
VARIABLES PRINCIPALES
a'
b'
c'
d'
e'
f'
g'
h'
u'
v'
w'
x'
y'
z'
a
b
c
d
e
f
g
h
u
v
w
x
y
z
Π=3.14592
0 # 2^-256
-1 # 20
-0.75 # 2^29

2. LA MEMORIA GRANDE .

Hemos hablado en I.2, en II.1.2 y en IV lo suficiente de la memoria grande para el empleo básico de las variables auxiliares. Este apéndice lo dedicamos a conocer más concretamente las distintas partes en vista a su mejor utilización, para facilitar determinadas operaciones manuales (como en el Apéndice 2), y para dar conocimiento exacto al programador de cómo está utilizando la memoria.

1.– Descripción física: La memoria grande está formada por dos tambores magnéticos, que se les llama Tambor 0 (Drum 0) y Tambor 1 (Drum 1). Cada uno tambores está dividido en ocho Zonas numeradas de 0 a 7 y cada zona consta de 32 sectores; los sectores están numerados de 0 a 511 a partir de la zona 0 del tambor 0 hasta la zona 7 del tambor 1. Por ejemplo los sectores de la zona 2 del tambor l serian los comprendidos entre el sector 320 y el sector 351.

Cada una de las zonas la podemos aislar mediante una llave de aislamiento que tiene el mismo número que la zona (ver fig. 6). Este aislamiento significa que no podemos transferir de la memoria rápida a la zona aislada de la grande, pero si podemos hacer la transferencia en sentido inverso. Con esto tenemos la seguridad de que la zona aislada no será destruida por un descuido de programación o de operación. No podemos aislar sectores individuales. En cada tambor existe un neón que se enciende cuando hace una transferencia y no se apaga hasta que efectúe una transferencia el otro tambor. Esto nos permite saber cual es el último de los tambores que ha operado.

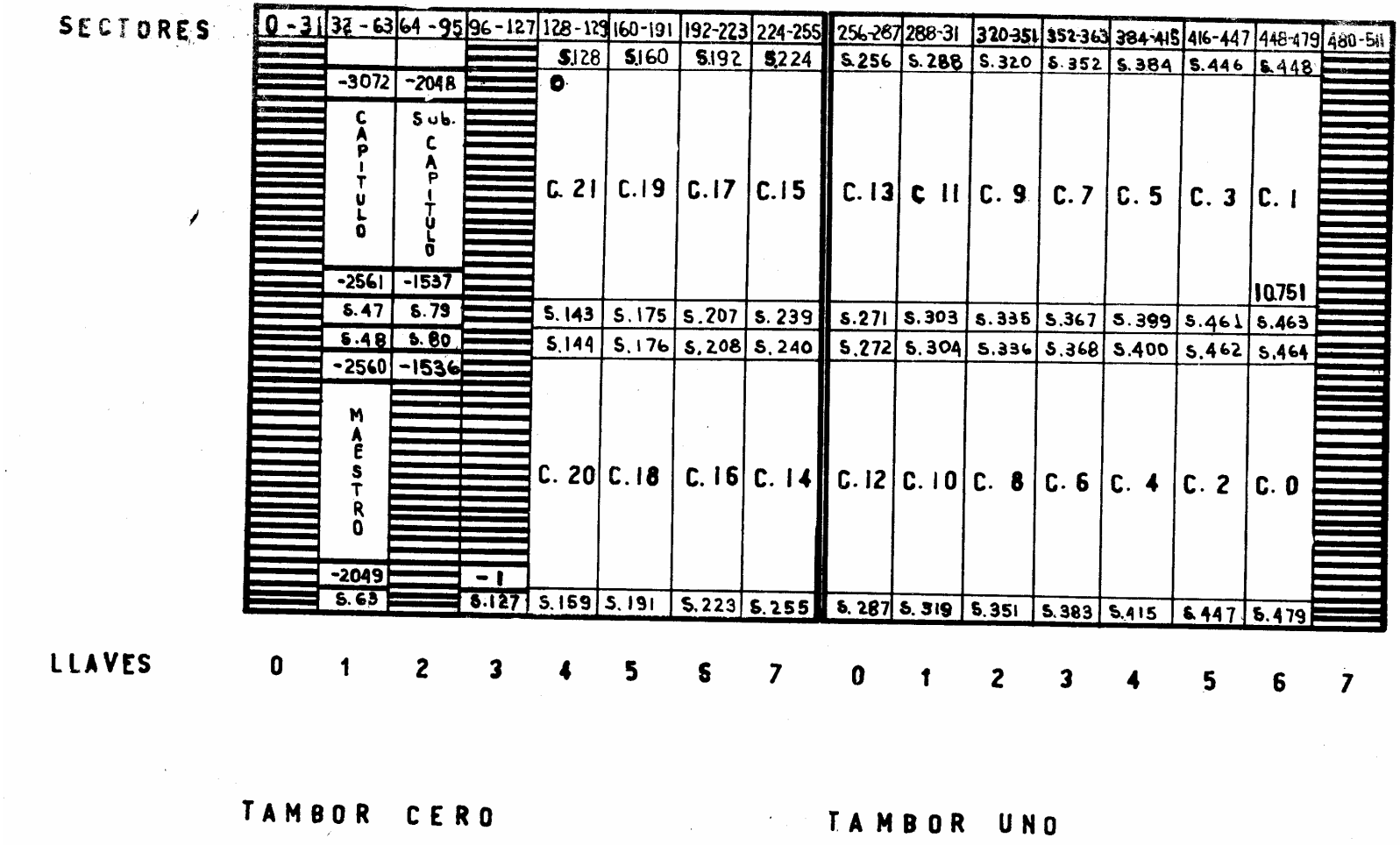
2.– Estructura funcional. Tres son las funciones principales de la memoria grande:

1.– Almacenar el programa traductor AUTOCODE INPUT.

2.– Almacenar los distintos capítulos del programa traducido.

3.– Almacenar números intermedios de nuestro proceso de cálculo.

2.1 El programa traductor consta de dos partes: una que se precisa solo durante la fase de traducción y otra que se precisa también en la ejecución (como son, por ejemplo, las subrutinas para el cálculo de las funciones elementales). La primera parte está almacenada en los sectores 80 a 127, y la segunda en los sectores 0 a 31, ocupando en total las zonas 0 y 3 y parte de la 2. Pueden considerarse como parte del AUTOCODE INPUT las rutinas correspondientes a las Facilidades para matrices, que se las almacena en los sectores 480 a 511.


SECTORES
0 - 31
32 - 63
64 - 95
96 - 127
128 - 129
160 - 191
192 - 223
224 - 255
256 - 267
288 - 31
320 - 351
352 - 363
384 - 415
416 - 447
448 - 479
480 - 511
$128
$160
$192
$224
$256
$288
$320
$352
$384
$446
$448
-3072
-2048
0
CAPITULO
Sub. CAPITULO
C. 21
C. 19
C. 17
C. 15
C. 13
C 11
C. 9
C. 7
C. 5
C. 3
C. 1
-2561
-1537
10751
S. 47
S. 79
S. 143
S. 175
S. 207
S. 239
S. 271
S. 303
S. 335
S. 367
S. 399
S. 461
S. 463
S. 48
S. 80
S. 144
S. 176
S. 208
S. 240
S. 272
S. 304
S. 336
S. 368
S. 400
S. 462
S. 464
-2560
-1536
MAESTRO
C. 20
C. 18
C. 16
C. 14
C. 12
C. 10
C. 8
C. 6
C. 4
C. 2
C. 0
-2049
-1
S. 63
S. 127
S. 159
S. 191
S. 223
S. 255
S. 287
S. 319
S. 351
S. 383
S. 415
S. 447
S. 479
LLAVES
0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
TAMBOR CERO
TAMBOR UNO

2.2 Los capítulos traducidos se almacenan en la memoria grande según el número que tienen asociado. Así al capitulo 0 le corresponden los sectores 464 a 479, al capitulo 1 los sectores 448 a 463, y al capítulo 21, máximo posible, los sectores 128 a 143. Independientemente del orden de lectura de los capítulos, se almacenarán siempre en el lugar correspondiente a su número. Dada la particularidad del capitulo 0 cuyo close ordena el comienzo de los cálculos, éste debe ser leído siempre al último. En los macro–programas que contengan varios programas, éstos pueden tener capítulos con el mismo número, entonces la ubicación de los capítulos del primer programa leído coincidirá con la anteriormente dada, y la del programa siguiente se hace considerando como capitulo 1 al siguiente al último leído del programa anterior. Pero no puede existir más de un capitulo 0 al que siempre corresponde la misma ubicación. El número máximo de capítulos que puede contener la memoria grande es de 22, incluyendo el capitulo cero.

2.3 Pero lo que principalmente nos interesa es conocer la ubicación de las variables auxiliares, para la mejor organización de nuestro programa, e impedir la destrucción de algún capitulo provocada por la superposición de números en él. Como ya sabemos las variables tienen direcciones numéricas que van de 0 a 10751 y de –l a –3072. Las variables 0 a 10751 ocupan los mismos lugares que los capítulos 21 a 1 (véase la fig.5). Las variables de –l a –3072 las dividimos en tres grupos según las zonas que invaden, el primero va de –l a –1536, que al ser utilizadas destruyen la primera parte del programa traductor a que hacíamos referencia en 2.1; de –1537 a –2048 no las debemos usar si utilizamos una instrucción preserve en el nivel de los subcapítulos (2º nivel); de –2049 a –2560 no la debemos utilizar si existe la instrucción preserve en el nivel de los capítulos (1º nivel); de –2561 a –3072 si utilizamos las facilidades para matrices, o tenemos en nuestro programa la instrucción r m p.

Estas últimas limitaciones son debidas a la existencia de tres depósitos, dos de ellos llamados DEPOSITO MAESTRO y DEPOSITO SUBCAPITULO que ocupan los sectores 48 a 63 y 64 a 79 , respectivamente; se utilizan para almacenar el contenido de la memoria actual de números ordenados por la instrucción preserve. El tercero llamado depósito especial sirve para depositar el contenido de la memoria actual de instrucciones durante la utilización de las facilidades para matrices, o mientras procedemos a una nueva traducción ordenada por la instrucción r m p.

La figura nos muestra en forma esquemática cada una de las partes principales de la memoria grande, y recomendamos su empleo para la organización de macroprogramas

que empleen varios programas y variables auxiliares, así como si se precisa utilizar las instrucciones down con preserve . La tabla I nos da el número de la primera variable auxiliar contenida en cada sector, y éstos agrupados por zonas. La utilidad de esta tabla es su referencia a las zonas que queramos aislar, o a los sectores que queramos imprimir utilizando los post−mortem u otras facilidades.

TABLA I

Las columnas impares indican el número de los sectores. Las columnas pares el número de la primera variable auxiliar del sector que se indica a su izquierda. Cada ocho filas constituyen una zona, cuyo número va indicado a su cabeza. Estos números coinciden con los de las llaves de aislación correspondientes.

TAMBOR 0

1

32 –3072	40 –2816	48 –2560	56 –2304
33 –3040	41 –2784	49 –2528	57 –2272
34 –3008	42 –2752	50 –2496	58 –2240
35 –2976	43 –2720	51 –2464	59 –2208
36 –2944	44 –2688	52 –2432	60 –2176
37 –2912	45 –2656	53 –2400	61 –2144
38 –2880	46 –2624	54 –2368	62 –2112
39 –2848	47 –2592	55 –2336	63 –2080

2

64 −2048	72 −1792	80 −1536	88 −1280
65 −2016	73 −1760	81 −1504	89 −1248
66 −1984	74 −1728	82 −1472	90 −1216
67 −1952	75 −1696	83 −1440	91 −1184
68 −1920	76 −1664	84 −1408	92 −1152
69 −1888	77 −1632	85 −1376	93 −1120
70 −1856	78 −1600	86 −1344	94 −1088
71 −1824	79 −1568	87 −1312	95 −1056

3

96 −1024	104 −768	112 −512	120 −256
97 −992	105 −736	113 −480	121 −224
98 −960	106 −704	114 −448	122 −192
99 −928	107 −672	115 −416	123 −160
100 −896	108 −640	116 −384	124 −128
101 −864	109 −608	117 −352	125 −96
102 −832	110 −576	118 −320	126 −64
103 −800	111 −544	119 −288	127 −32

4

128	0	136	256	144	512	152	768
129	32	137	288	145	544	153	800
130	64	138	320	146	576	154	832
131	96	139	352	147	608	155	864
132	128	140	384	148	640	156	896
133	160	141	416	149	672	157	928
134	192	142	448	150	704	158	960
135	224	143	480	151	736	159	992

5

160	1024	168	1280	176	1536	184	1792
161	1056	169	1312	177	1568	185	1824
162	1088	170	1344	178	1600	186	1856
163	1120	171	1376	179	1632	187	1888
164	1152	172	1408	180	1664	188	1920
165	1184	173	1440	181	1696	189	1952
166	1216	174	1472	182	1728	190	1984
167	1248	175	1504	183	1760	191	2016

6

192	2048	200	2304	208	2560	216	2816
193	2080	201	2336	209	2592	217	2848
194	2112	202	2368	210	2624	218	2880
195	2144	203	2400	211	2656	219	2912
196	2176	304	2432	212	2688	220	2944
197	2208	205	2464	213	2720	221	2976
198	2240	206	2496	214	2752	222	3008
199	2272	207	2528	215	2784	223	3040

7

224	3072	232	3328	240	3584	248	3840
225	3104	233	3360	241	3616	249	3872
226	3136	234	3392	242	3648	250	3904
227	3168	235	3424	243	3680	251	3936
228	3200	236	3456	244	3712	252	3968
229	3232	237	3488	245	3744	253	4000
230	3264	238	3520	246	3776	254	4032
231	3296	239	3552	247	3808	255	4064

TAMBOR 1

0

256	4096	264	4352	273	4608	280	4864
257	4128	265	4384	273	4640	281	4896
258	4160	266	4416	274	4672	282	4928
259	4192	267	4448	275	4704	283	4960
260	4224	268	4480	276	4736	284	4992
261	4256	269	4512	277	4768	285	5024
262	4288	270	4544	278	4800	286	5056
263	4320	271	4576	279	4832	287	5088

1

288	5120	296	5376	304	5632	312	5888
289	5152	297	5408	305	5664	313	5920
290	5184	298	5440	306	5696	314	5952
291	216	299	5472	307	5728	315	5984
292	5248	300	5504	308	5760	316	6016
293	5280	301	5536	309	5792	317	6048
294	5312	302	5568	310	5824	318	6080
295	5344	303	5600	311	5856	319	6112

2

320	6144	328	6400	336	6656	344	6912
321	6176	329	6432	337	6688	345	6944
322	6208	330	6464	338	6720	346	6976
323	6240	331	6496	339	6752	347	7008
324	6272	332	6528	340	6784	348	7040
325	6304	333	6560	341	6816	349	7072
326	6336	334	6592	342	6848	350	7104.
327	6368	335	6624	343	6880	351	7136

3

252	7168	360	7424	368	7680	376	7936
353	7200	361	7456	369	7712	377	7968
354	7232	362	7488	370	7744	378	8000
355	7264	363	7520	371	7776	379	8032
356	7296	364	7552	372	7808	380	8064
357	7328	365	7584	373	7840	381	8096
358	7360	366	7616	374	7872	382	8128
359	7392	367	7648	375	7904	383	8160

4

384	8192	392	8448	400	8704	408	8960
385	8224	393	8480	401	8736	409	8992
386	8256	394	8512	402	8768	410	9024
387	8288	395	8544	403	8800	411	9056
388	8320	396	8576	404	8832	412	9088
389	8352	397	8608	405	8864	413	9120
390	8384	398	8640	406	8896	414	9152
391	8416	399	8672	407	8928	415	9184

5

416	9216	424	9472	432	9728	440	9984
417	9248	425	9504	433	9760	441	10016
418	9280	426	9536	434	9792	442	10048
419	9312	427	9568	435	9824	443	10080
420	9344	428	9600	436	9856	444	10112
421	9376	429	9633	437	9888	445	10144
422	9408	430	9664	438	9920	446	10176
423	9440	431	9696	439	9952	447	10208

6

448	10240	456	10496
449	10272	457	10528
450	10304	458	10560
451	10336	459	10592
452	10368	460	10624
453	10400	461	10656
454	10432	462	10688
455	10464	463	10720

APENDICE 2

INSTRUCCIONES DE CONVENCIONAL ACEPTADAS POR AUTOCODE

Para aumentar su flexibilidad, el lenguaje AUTOCODE tiene la posibilidad de aceptar instrucciones escritas en lenguaje CONVENCIONAL (*) con algunas variantes que vamos a describir haciendo distinción según sean las direcciones que empleemos escritas en convencional o en autocode (variables, índices o marcas).

1. Direcciones en autocode. Si el operando es una variable, un índice o una marca de las utilizadas en el sistema autocode, las instrucciones en convencional tomarían la forma :

FB (α)	α= cualquier variable principal o especial o una constante
FB (i)	i = un índice o un entero
FB (m)	m = una marca

En el primer caso se trata de las operaciones con números largos (40 bits), indicadas en el código de convencional por los números 40 a 45 y 50 a 55. Cuando se utilicen variables con índice variable (como x_i, $y_{(j+1)}$), el B−dígito debe ser cero.
En el segundo caso los códigos aplicables son los indicados por los números 00 a 07 y 20 a 27.
En el tercer caso la función puede ser cualquiera de las de salto.

Ejemplos :

a) Son posibles las siguientes expresiones
411 (x)
422 (z_s)
510 (x_i)
200 (k)

(*) Estas notas son útiles para quienes conocen ya el lenguaje convencional.
Recordaremos que la forma de la instrucción era

F-B D

La F expresada mediante dos dígitos decimales representa la función a ejecutar, B indicado mediante un solo dígito decimal expresa el registro B que interviene en la instrucción, y D (que puede ser representado de diversas formas) es la dirección del operando.

b) Para calcular e *el* valor del polinomio

$$y = a_0 x^{10} + a_1 x^{9} + \ldots\ldots + a_{10}$$

procederemos

400 (a_0)
i = 1(1) 10
500 (x)
420 (a_i)

repeat
410 (y)

2. Direcciones en convencional. La parte de instrucción expresada entre paréntesis del párrafo anterior puede indicar únicamente direcciones, y por tanto no se puede aplicar el anterior convenio para las instrucciones en que esos dígitos indican una constante; el siguiente convenio se puede aplicar a cualquier tipo de instrucción:

FB, α α= entero o dirección absoluta,.

Se pueden utilizar: la forma de dirección página–registro (2.16 *+)* y los símbolos ≠ e = . No son permitidas las formas de direcciones flotantes o relativas.

Ejemplos :

a) Son posibles
321, 28
400, 2
512, 2.16
590, 1.0

No son permitidas
400, v1
430, 2x1

b) Para el cálculo del polinomio:

$$y = a_0 x^{10} + a_1 x^{9} + \ldots\ldots + a_{10}$$

procederíamos :

400 (a0)
300, —*9*
1) 500 (x)
427 (a10)
380 (i)
410 (u)

3. Utilización de las facilidades de convencional para la salida. Una de las importantes aplicaciones es la flexibilidad que podemos lograr en la salida utilizando la instrucción 620, n. Esta instrucción tiene por efecto perforar en la cinta de resultados el carácter indicado por n. (véase tabla II), donde n indica el valor en binario que corresponde a cada perforación, y que es interpretado de acuerdo a la tabla adjunta. Es importante tener presente el significado de los caracteres 0 (FS) y 27 (LS) pues necesitaremos perforarlos delante de cada grupo de números (o signos especiales) o de cada grupo de letras.

Ejemplo: Si queremos escribir durante nuestro proceso los resultados de la siguiente manera

CASO N
al = valor correspondiente a2 = valor correspondiente

pondríamos :

Ver página siguiente

```
newline
620,27   siguen letras
620,3      c
620,1      a
620,19     s
620,15     o
620,0    siguen cifras
620,14   espacio
620,14   espacio

print (u) 2,0
newline
620,27   siguen letras
620,1      a
620,0    siguen números
620,1      1
620,10     =
print (a1)2,4
620,27   siguen letras
620,1      a
620,0    siguen números
620,2      2
620,10     =
print (a2) 2,4
```

CINTA	VALOR	IMPRESORA	
		CIFRAS	LETRAS
.	0	FIG.	SHIFT.
. ●	1	1	A
.●	2	2	B
.●●	3	*	C
●.	4	4	D
●. ●	5	(	E
●.●	6	)	F
●.●●	7	7	G
● .	8	8	H
● . ●	9	≠	I
● .●	10	=	J
● .●●	11	-	K
●●.	12	v	L
●●. ●	13	LF	M
●●.●	14	SP	N
●●.●●	15	,	O
● .	16	0	P
● . ●	17	>	Q
● .●	18	>	R
● .●●	19	3	S
● ●.	20	→	T
● ●. ●	21	5	U
● ●.●	22	6	V
● ●.●●	23	/	W
●● .	24	x	X
●● . ●	25	9	Y
●● .●	26	+	Z
●● .●●	27	LET.	SHIFT.
●●●.	28	.	.
●●●. ●	29	n	?
●●●.●	30	CR	£
●●●.●●	31	ER	ER

TABLA II

S U M A R I O

Impreso en los talleres del Departamento de Biblioteca y Publicaciones de la Facultad de Ciencias Exactas y Naturales.